IntechOpen

Tools in Artificial Intelligence

Edited by Paula Fritzsche

Tools in Artificial Intelligence

Edited by
Paula Fritzsche

Tools in Artificial Intelligence
http://dx.doi.org/10.5772/74
Edited by Paula Fritzsche

First published in Croatia, 2008 by INTECH d.o.o.
eBook (PDF) Published by IN TECH d.o.o.
Place and year of publication of eBook (PDF): Rijeka, 2019.
IntechOpen is the global imprint of IN TECH d.o.o.
Printed in Croatia

Legal deposit, Croatia: National and University Library in Zagreb

Additional hard and PDF copies can be obtained from orders@intechopen.com

Tools in Artificial Intelligence
Edited by Paula Fritzsche

p. cm.

ISBN 978-953-7619-03-9

eBook (PDF) ISBN 978-953-51-5794-6

Preface

Artificial Intelligence (AI) is often referred to as a branch of science which deals with helping machines find solutions to complex problems in a more human-like fashion. It is generally associated with Computer Science, but it has many important links with other fields such as Maths, Psychology, Cognition, Biology and Philosophy. The AI success is due to its technology has diffused into everyday life. Neural networks, fuzzy controls, decision trees and rule-based systems are already in our mobile phones, washing machines and business applications.

The book "Tools in Artificial Intelligence" offers in 27 chapters a collection of all the technical aspects of specifying, developing, and evaluating the theoretical underpinnings and applied mechanisms of AI tools. Topics covered include neural networks, fuzzy controls, decision trees, rule-based systems, data mining, genetic algorithm and agent systems, among many others.

The goal of this book is to show some potential applications and give a partial picture of the current state-of-the-art of AI. Also, it is useful to inspire some future research ideas by identifying potential research directions. It is dedicated to students, researchers and practitioners in this area or in related fields.

Editor

Paula Fritzsche
Computer Architecture and Operating Systems Department
University Autonoma of Barcelona
Spain
e-mail: paula.fritzsche@caos.uab.es

Contents

Computational Intelligence in Software Cost Estimation: Evolving Conditional Sets of Effort Value Ranges

Efi Papatheocharous and Andreas S. Andreou
Department of Computer Science, University of Cyprus, Cyprus

1. Introduction

In the area of software engineering a critical task is to accurately estimate the overall project costs for the completion of a new software project and efficiently allocate the resources throughout the project schedule. The numerous software cost estimation approaches proposed are closely related to cost modeling and recognize the increasing need for successful project management, planning and accurate cost prediction. Cost estimators are continually faced with problems stemming from the dynamic nature of the project development process itself. Software development is considered an intractable procedure and inevitably depends highly on several complex factors (e.g., specification of the system, technology shifting, communication, etc.). Normally, software cost estimates increase proportionally to development complexity rising, whereas it is especially hard to predict and manage the actual related costs. Even for well-structured and planned approaches to software development, cost estimates are still difficult to make and will probably concern project managers long before the problem is adequately solved.

During a system's life-cycle, one of the most important tasks is to effectively describe the necessary development activities and estimate the corresponding costs. This estimation, once successful, allows software engineers to optimize the development process, improve administration and control over the project resources, reduce the risks caused by contingencies and minimize project failures (Lederer & Prasad, 1992). Subsequently, a commonly investigated approach is to accurately estimate some of the fundamental characteristics related to cost, such as effort and schedule, and identify their inter-associations. Software cost estimation is affected by multiple parameters related to technologies, scheduling, manager and team member skills and experiences, mentality and culture, team cohesion, productivity, project size, complexity, reliability, quality and many more. These parameters drive software development costs either positively or negatively and are considerably very hard to measure and manage, especially at an early project development phase. Hence, software cost estimation involves the overall assessment of these parameters, even though for the majority of the projects, the most dominant and popular metric is the effort cost, typically measured in person-months.

Recent attempts have investigated the potential of employing Artificial Intelligence-oriented methods to forecast software development effort, usually utilising publicly available datasets (e.g., Dolado, 2001; Idri et al., 2002; Jun & Lee, 2001; Khoshgoftaar et al., 1998; Xu & Khoshgoftaar, 2004) that contain a wide variety of cost drivers. However, these cost drivers are often ambiguous because they present high variations in both their measure and values. As a result, cost assessments based on these drivers are somewhat unreliable. Therefore, by detecting those project cost attributes that decisively influence the course of software costs and similarly define their possible values may constitute the basis for yielding better cost estimates. Specifically, the complicated problem of software cost estimation may be reduced or decomposed into devising and evolving bounds of value ranges for the attributes involved in cost estimation using the theory of conditional sets (Packard, 1990). These ranges may then be used to attain adequate predictions in relation to the effort located in the actual project data. The motivation behind this work is the utilization of rich empirical data series of software project cost attributes (despite suffering from limited quality and homogeneity) to produce robust effort estimations. Previous work on the topic has suggested high sensitivity to the type of attributes used as inputs in a certain Neural Network model (MacDonell & Shepperd, 2003). These inputs are usually discrete values from well-known and publicly available datasets. The data series indicate high variations in the attributes or factors considered when estimating effort (Dolado, 2001). The hypothesis is that if we manage to reduce the sensitivity of the technique by considering indistinct values in terms of ranges, instead of crisp discrete values, and if we employ an evolutionary technique, like Genetic Algorithms, we may be able to address the effect of attribute variations and thus provide a near-to-optimum solution to the problem. Consequently, the technique proposed in this chapter may provide some insight regarding which cost drivers are the most important. In addition, it may lead to identifying the most favorable attribute value ranges for a given dataset that can yield a 'secure' and more flexible effort estimate, again having the same reasoning in terms of ranges. Once satisfactory and robust value ranges are detected and some confidence regarding the most influential attributes is achieved, then cost estimation accuracy may be improved and more reliable estimations may be produced.

The remainder of this work is structured as follows: Section 2 presents a brief overview of the related software cost estimation literature and mainly summarizes Artificial Intelligence techniques, such as Genetic Algorithms (GA) exploited in software cost estimation. Section 3 encompasses the description of the proposed methodology, along with the GA variance constituting the method suggested, a description of the data used and the detailed framework of our approach. Consequently, Section 4 describes the experimental procedure and the results obtained after training and validating the genetic evolution of value ranges for the problem of software cost estimation. Finally, Section 5 concludes the chapter with a discussion on the difficulties and trade-offs presented by the methodology in addition to suggestions for improvements in future research steps.

2. Related work

Traditional model-based approaches to cost estimation, such as COCOMO, Function Point Analysis (FPA) and SLIM, assume that if we use some independent variables (i.e., project characteristics) as inputs and a dependent variable as the output (namely development effort), the resulted complex I/O relationships may be captured by a formula (Pendharkar et al., 2005). In reality, this is never the case. In COCOMO (Boehm, 1981), one of the most popular models for software cost estimation, the development effort is calculated using the estimated delivered source instructions and an effort adjustment factor, applied to three

distinct levels (basic, intermediate and advanced) and two constant parameters. COCOMO was revised in newer editions (Boehm et al., 1995; Boehm et al., 2000), using software size as the primary factor and 17 secondary cost factors. The revised model is regression-based and involves a mixture of three cost models, each corresponding to a stage in the software life-cycle namely: Applications Composition, Early Design and Post Architecture. The Application Composition stage involves prototyping efforts; the Early Design stage includes only a small number of cost drivers as there is not enough information available at this point to support fine-grained cost estimation; the Post Architecture stage is typically applied after the software architecture has been defined and provides estimates for the entire development life-cycle using effort multipliers and exponential scale factors to adjust for project, platform, personnel, and product characteristics.
Models based on Function Points Analysis (FPA) (Albrecht & Gaffney, 1983) mainly involve identifying and classifying the major system components such as external inputs, external outputs, logical internal files, external interface files and external inquiries. The classification is based on their characterization as 'simple', 'average' or 'complex', depending on the number of interacting data elements and other factors. Then, the unadjusted function points are calculated using a weighting schema and adjusting the estimations utilizing a complexity adjustment factor. This is influenced by several project characteristics, namely data communications, distributed processing, performance objective, configuration load, transaction rate, on-line data entry, end-user efficiency, on-line update, complex processing, reusability, installation ease, operational ease, multiple sites and change facilitation.
In SLIM (Fairley, 1992) two equations are used: the software productivity level and the manpower equation, utilising the Rayleigh distribution (Putnam & Myers, 1992) to estimate project effort schedule and defect rate. The model uses a stepwise approach and in order to be applicable the necessary parameters must be known upfront, such as the system size - measured in KDSI (thousand delivered source instructions), the manpower acceleration and the technology factor, for which different values are represented by varying factors such as hardware constraints, personnel experience and programming experience. Despite being the forerunner of many research activities, the traditional models mentioned above, did not produce the best possible results. Even though many existing software cost estimation models rely on the suggestion that predictions of a dependent variable can be formulated if several (in)dependent project characteristics are known, they are neither a silver bullet nor the best-suited approaches for software cost estimation (Shukla, 2000).
Over the last years, computational intelligence methods have been used attaining promising results in software cost estimation, including Neural Networks (NN) (Jun & Lee, 2001; Papatheocharous & Andreou, 2007; Tadayon, 2005), Fuzzy Logic (Idri et al., 2002; Xu & Khoshgoftaar , 2004), Case Based Reasoning (CBR) (Finnie et al., 1997; Shepperd et al., 1996), Rule Induction (RI) (Mair et al., 2000) and Evolutionary Algorithms.
A variety of methods, usually evolved into hybrid models, have been used mainly to predict software development effort and analyze various aspects of the problem. Genetic Programming (GP) is reported in literature to provide promising approximations to the problem. In (Burgess & Leftley, 2001) a comparative evaluation of several techniques is performed to test the hypothesis of whether GP can improve software effort estimates. In terms of accuracy, GP was found more accurate than other techniques, but does not converge to a good solution as consistently as NN. This suggests that more work is needed towards defining which measures, or combination of measures, is more appropriate for the

particular problem. In (Dolado, 2001) GP evolving tree structures, which represent software cost estimation equations, is investigated in relation to other classical equations, like the linear, power, quadratic, etc. Different datasets were used in that study yielding diverse results, classified as 'acceptable', 'moderately good', 'moderate' and 'bad' results. Due to the reason that the datasets examined varied extremely in terms of complexity, size, homogeneity, or values' granularity consistent results were hard to obtain. In (Lefley, & Shepperd 2003) the use of GP and other techniques was attempted to model and estimate software project effort. The problem was modeled as a symbolic regression problem to offer a solution to the problem of software cost estimation and improve effort predictions. The so-called "Finnish data set" collected by the software project management consultancy organization SSTF was used in the context of within and beyond a specific company and obtained estimations that indicated that with the approaches of Least-Square Regression, NN and GP better predictions could be obtained. The results from the top five percent estimators yielded satisfactory performance in terms of Mean Relative Error (MRE) with the GP appearing to be a stronger estimator achieving better predictions, closer to the actual values more often than the rest of the techniques. In the work of (Huang & Chiu, 2006) a GA was adopted to determine the appropriate weighted similarity measures of effort drivers in analogy-based software effort estimation models. These models identify and compare the software project developed with similar historical projects and produce an effort estimate. The ISBSG and the IBM DP services databases were used in the experiments and the results obtained showed that among the applied methods, the GA produced better estimates and the method could provide objective weights for software effort drivers rather than the subjective weights assigned by experts.

In summary, software cost estimation is a complicated activity since there are numerous cost drivers, displaying more than a few value discrepancies between them, and highly affecting development cost assessment. Software development metrics for a project reflect both qualitative measures, such as, team experiences and skills, development environment, group dynamics, culture, and quantitative measures, for example, project size, product characteristics and available resources. However, for every project characteristic the data is vague, dissimilar and ambiguous, while at the same time formal guidelines on how to determine the actual effort required to complete a project based on specific characteristics or attributes do not exist. Previous attempts to identify possible methods to accurately estimate development effort were not as successful as desired, mainly because calculations were based on certain project attributes of publicly available datasets (Jun & Lee, 2001). Nevertheless, the proportion of evaluation methods employing historical data is around 55% from a total of 304 research papers investigated by Jorgensen & Shepperd in 2004 (Jorgensen & Shepperd, 2007). According to the same study, evaluation of estimation methods requires that the datasets be as representative as possible to the current or future projects under evaluation. Thus, if we wish to evaluate a set of projects, we might consider going a step back, and re-define a more useful dataset in terms of conditional value ranges. These ranges may thus lead to identifying representative bounds for the available values of cost drivers that constitute the basis for estimating average cost values.

3. The proposed cost estimation framework

The framework proposed in this chapter encompasses the application of the theory of conditional sets in combination with Genetic Algorithms (GAs). The idea is inspired by the

work presented by Packard et al. (Meyer & Packard, 1992; Packard, 1990) utilising GAs to evolve conditional sets. The term conditional set refers to a set of boundary conditions. The main concept is to evaluate the evolved value ranges (or conditional sets) and extract underlying determinant relationships among attributes and effort in a given dataseries. This entails exploring a vast space of solutions, expressed in ranges, utilising additional manufactured data than those located into a well-known database regularly exploited for software effort estimation.
What we actually propose is a method for investigating the prospect of identifying the exact value ranges for the attributes of software projects and determining the factors that may influence development effort. The approach proposed implies that the attributes' value ranges and corresponding effort value ranges are automatically generated, evaluated and evolved through selection and survival of the fittest in a way similar to natural evolution (Koza, 1992). The goal is to provide complementing weights (representing the notion of ranked importance to the associated attributes) together with effort predictions, which could possibly result in a solution more efficient and practical than the ones created by other models and software cost estimation approaches.

3.1 Conditional sets theory and software cost

In this section we present some definitions and notations of conditional sets theory in relation to software cost based on paradigms described in (Adamopoulos et al., 1998; Packard, 1990).
Consider a set of *n* cost attributes *{A_1, A_2,…, A_n}*, where each A_i has a corresponding discrete value x_i. A software project may be described by a vector of the form:

$$L = \{x_1, x_2, \ldots, x_n\} \tag{1}$$

Let us consider a condition C_i of the form:

$$C_i : (lb_i < x_i < ub_i),\ i = 1 \ldots n \tag{2a}$$

where lb_i and ub_i are the lower and upper bounds of C_i respectively for which:

$$\forall C_i : |lb_i - ub_i| < \varepsilon \tag{2b}$$

that is, lb_i and ub_i have minimal difference in their value, under a specific threshold ε.
Consider also a conditional set *S*; we say that *S* is of length *l* (≤*n*) if it entails *l* conditions of the form described by equations (2a) and (2b), which are coupled via the logical operators of AND and OR as follows:

$$S_{AND} = C_1 \wedge C_2 \wedge \ldots \wedge C_l \tag{3}$$

$$S_{OR} = C_1 \vee C_2 \vee \ldots \vee C_l \tag{4}$$

We consider each conditional set *S* as an individual in the population of our GA, which will be thoroughly explained in the next section as part of the proposed methodology. We use equations (3) and (4) to describe conditional sets representing cost attributes, or to be more precise, cost metrics. What we are interested in is the definition of a set of software projects, *M*, the elements of which are vectors as in equation (1) that hold the values of the specific cost attributes used in relation with a conditional set. More specifically, the set *M* can be defined as follows:

$$M = \{L_1, L_2, ..., L_m\} \tag{5}$$

$$L_i = \{x_{i,1}, x_{i,2}, ..., x_{i,l}\}, \ i = 1...m \tag{6}$$

where *l* denotes the number of cost attributes of interest.
A conditional set *S* is related to *M* according to the conditions in equations (3) or (4) that are satisfied as follows:

$$\forall L_i : \quad x_{i,k} \ satisfies \ C_k \ , \ i = 1...m, \ \ k = 1...l \ \text{(AND)} \tag{7}$$

$$\begin{gathered} x_{i,1} \ satisfies \ C_1 \ OR \ x_{i,2} \ satisfies \ C_2, ... \\ ..., \ OR \ x_{i,l} \ satisfies \ C_l, \ \ i = 1...m, \ \text{(OR)} \end{gathered} \tag{8}$$

3.2 Methodology

Before proceeding to describe the methodology proposed we provide a short description of the dataset used. The dataset was obtained from the International Software Benchmarking Standards Group (ISBSG, Repository Data Release 9 - ISBSG/R9, 2005) and contains an analysis of software project costs for a group of projects. The projects come from a broad cross section of industry and range in size, effort, platform, language and development technique data. The release of the dataset used contains 92 variables for each of the projects and hosts multi-organizational, multi-application domain and multi-environment data that may be considered fairly heterogeneous (International Software Benchmarking Standards Group, http://www.isbsg.org/). The dataset was recorded following data collection standards ensuring broad acceptance. Nevertheless, it contains more than 4,000 data from more than 20 countries and hence it is considered highly heterogeneous. Therefore, data acquisition, investigation and employment of the factors that impact planning, management and benchmarking of software development projects should be performed very cautiously.
The proposed methodology is divided into three steps, namely the data pre-processing step, the application of the GA and the evaluation of the results. Figure 1 summarizes the methodology proposed and the steps followed for evolving conditional sets and providing effort range predictions. Several filtered sub-sets of the ISBSG/R9 dataset were utilized for the evolution of conditional sets, initially setting up the required conditional sets. The conditional sets are coupled with two logical operators (AND and OR) and the investigation lies with extracting the ranges of project features or characteristics that describe the associated project effort. Furthermore, the algorithm creates a random set or initial population of conditions (individuals). The individuals are then evolved through specific genetic operators and evaluated internally using the fitness functions. The evolution of individuals continues while the termination criteria are not satisfied, among these a maximum number of iterations (called generations or epochs) or no improvement in the maximum fitness value occurs for a specific number of generations. The top 5% individuals resulting in the higher fitness evaluations are accumulated into the optimum range

population, which then are advanced to the next algorithm generation (repetition). At the end, the final population produced that satisfies the criteria is used to estimate the mean effort, whereas at the evaluation step, the methodology is assessed through various performance metrics. The most successful conditional sets evolved by the GA that have small assembled effort ranges with relatively small deviation from the mean effort, may then be used to predict effort of new, unknown projects.

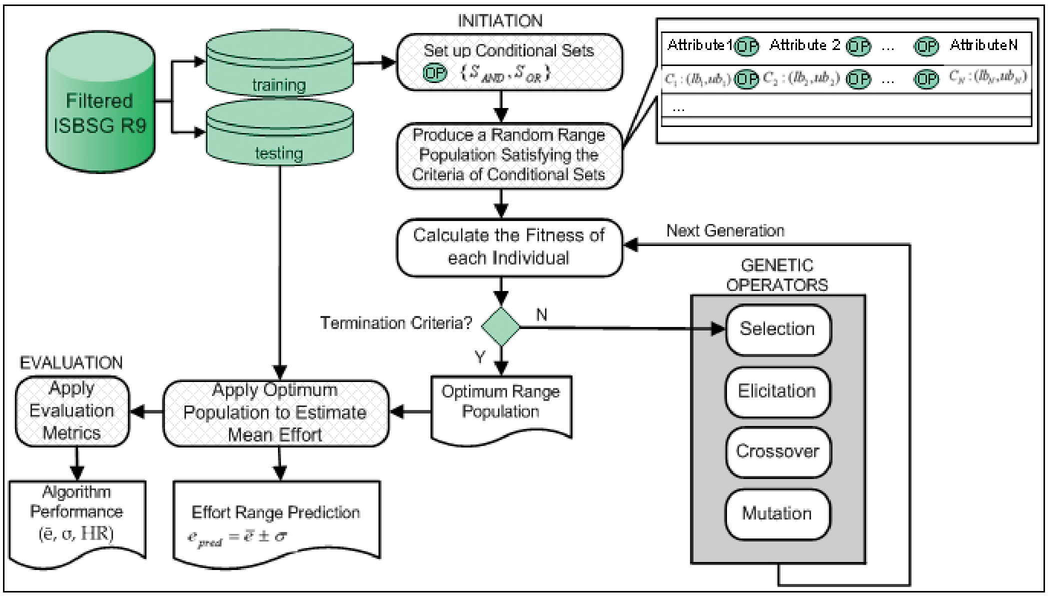

INITIATION
Filtered ISBSG R9
training
testing
Set up Conditional Sets
OP $\{S_{AND}, S_{OR}\}$
Produce a Random Range Population Satisfying the Criteria of Conditional Sets
Attribute 1 OP Attribute 2 OP ... OP Attribute N
$C_1 : (lb_1, ub_1)$ OP $C_2 : (lb_2, ub_2)$ OP ... OP $C_N : (lb_N, ub_N)$
...
Calculate the Fitness of each Individual
Next Generation
GENETIC OPERATORS
Termination Criteria?
N
Y
Selection
Elicitation
Crossover
Mutation
EVALUATION
Apply Evaluation Metrics
Apply Optimum Population to Estimate Mean Effort
Optimum Range Population
Algorithm Performance (ē, σ, HR)
Effort Range Prediction
$e_{pred} = \bar{e} \pm \sigma$

Fig. 1. Methodology followed for evolving conditional sets

3.2.1 Data pre-processing

In this step the most valuable set of attributes, in terms of contribution to effort estimation, are assembled from the original ISBSG/R9 dataset. After careful consideration of guidelines provided by the ISBSG and other research organizations, we decided to the formation of a reduced ISBSG dataset including the following main attributes: the project id (ID), the adjusted function points of the product (AFP), the project's elapsed time (PET), the project's inactive time (PIT), the project's delivery rate (productivity) in functional size units (PDRU), the average team size working on the project (ATS), the development type (DT), the application type (AT), the development platform (DP), the language type (LT), the primary programming language (PPL) and the resource level (RL) and the work effort expensed during the full development life-cycle (EFF) which will be used as a sort of output by the corresponding evolutionary algorithm. The attributes selected from the original, wider pool of ISBSG, were further filtered to remove those attributes with categorical-type data and other attributes that could not be included in the experimentation. Also, some attributes underwent value transformations, for example instead of PET and PIT we used their subtraction, normalized values for AFP and specific percentiles defining acceptance thresholds for filtering the data.
The first experiments following our approach indicated that further processing of the attributes should be performed, as the approach was quite strict and not applicable for heterogeneous datasets containing many project attributes with high deviations in their

values and measurement. Therefore, this led us to examine smaller, more compact, homogeneous and free from outlier subsets. In fact, we managed to extract three final datasets which we used in our final series of experiments. The first dataset (DS-1) contained the main attributes suggested by Function Point Analysis (FPA) to provide measurement of project software size, and included: Adjusted Function Points (AFP), Enquiry Count (EC), File Count (FC), Added Count (AC) and Changed Count (CC). These attributes were selected based on previous findings that considered them to be more successful in describing development effort after applying sensitivity analysis on the inputs with Neural Networks (Papatheocharous & Andreou, 2007). The second dataset (DS-2) is a variation of the previous dataset based on the preliminary results of DS-1, after performing normalization and removing the outliers according to the lower and upper thresholds defined by the effort box-plots. This resulted to the selection of the attributes: Normalized PDR-AFP (NAFP), Enquiry Count (EC), File Count (FC) and Added Count (AC). Finally, the third dataset (DS-3) created included the project attributes that can be measured early in the software life-cycle consisting of: Adjusted Function Points (AFP), Project's Delivery Rate (PDRU), Project's Elapsed Time (PET), Resource Level (RL) and Average Team Size (ATS) attributes in which also box-plots and percentile thresholds were used to remove outliers.

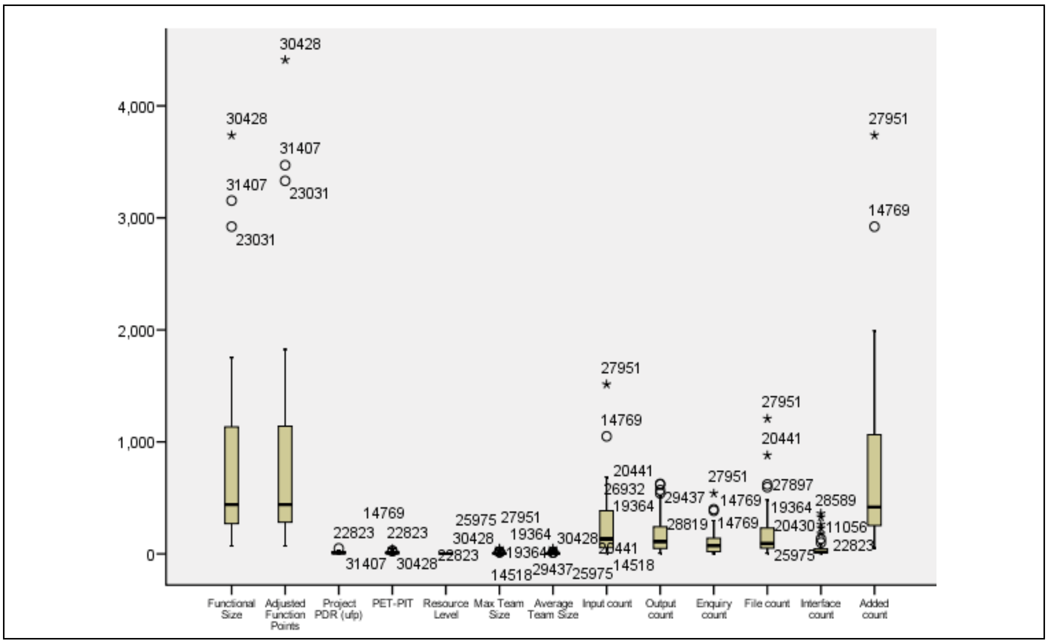


Fig. 2. Example of box-plots for the ISBSG project attributes (original full dataset)

It is noteworthy that each dataset also contained the values of the development work effort (EFF), the output attribute that we wanted to predict. As we already mentioned, the last data pre-processing step of the three datasets constructed included the cleaning of null and outlying values. The theory of box-plots was used to locate the outlying figures from the datasets and project cleaning was performed for each project variable separately. Figure 2 above shows an example of the box-plots created for each variable on the original full dataset. We decided to disregard the extreme outliers (marked as asterisks) occurring in each of the selected attributes and also exclude those projects considered as mild outliers (marked as circles), thus imposing more strict filtering associated with the output variable effort (EFF).

3.2.2 Genetic algorithm application

Genetic Algorithms (GAs) are evolutionary computational approaches that are domain-independent, and aim to find approximated solutions in complex optimization and search problems (Holland, 1992). They achieve this by pruning a population of individuals based on the Darwinian principle of reproduction and 'survival of the fittest' (Koza, 1992). The fitness of each individual is based on the quality of the simulated individual in the environment of the problem investigated. The process is characterized by the fact that the solution is achieved by means of a cycle of generations of candidate solutions that are pruned by using a set of biologically inspired operators. According to evolutionary theories, only the most suited solutions in a population are likely to survive and generate offspring, and transmit their biological heredity to the new generations. Thus, GAs are much superior to conventional search and optimization techniques in high-dimensional problem spaces due to their inherent parallelism and directed stochastic search implemented by recombination operators. The basic process of our GA operates through a simple cycle of three stages, as these were initially described by (Michalewicz, 1994):

Stage 1: Randomly create an initial population of individuals *P*, which represent solutions to the given problem (in our case, ranges of values in the form of equations (3) or (4)).

Stage 2: Perform the following steps for each generation:

- 2.1. Evaluate the fitness of each individual in the population using equations (9) or (10) below, and isolate the best individual(s) of all preceding populations.
- 2.2. Create a new population by applying the following genetic operators:
  - 2.2.1. *Selection*; based on the fitness select a subset of the current population for reproduction by applying the roulette wheel method. This method of reproduction allocates offspring values using a roulette wheel with slots sized according to the fitness of the evaluated individuals. It is a way of selecting members from a population of individuals in a natural way, proportional to the probability set by the fitness of the parents. The higher the fitness of the individual is, the greater the chance it will be selected, however it is not guaranteed that the fittest member goes to the next generation. So, additionally, elitism is applied, where the top best performing individuals are copied in the next generation and thus, rapidly increase the performance of the algorithm.
  - 2.2.2. *Crossover*; two or more individuals are randomly chosen from the population and parts of their genetic information are recombined to produce new individuals. Crossover with two individuals takes place either by exchanging their ranges at the crossover point (inter-crossover) or by swapping the upper or lower bound of a specific range (intra-crossover). The crossover takes place on one (or more) randomly chosen crossover point(s) along the structures of the two individuals.
  - 2.2.3. *Mutation*; randomly selected individuals are altered randomly and inserted into the new population. The alteration takes place at the upper or lower bound of a randomly selected range by adding or subtracting a small random number. Mutation intends to preserve the diversity of the population by expanding the search space into regions that may contain better solutions.
- 2.3. Replace the current population with the newly formed population.

Stage 3: Repeat from stage 2 unless a termination condition is satisfied. Output the individual with the best fitness as the near to optimum solution.

Each loop of the steps is called a generation. The entire set of iterations from population initialization to termination is called a run. At the termination of the process the algorithm promotes the "best-of-run" individual.

3.2.3 Evaluation

The individuals evolved by the GA are evaluated according to the newly devised fitness functions of AND or OR, specified as:

$$F_{AND} = k + \frac{1}{\sigma} + \frac{1}{\left(\sum_{i=1}^{l}(ub_i - lb_i) * w_i\right)} \tag{9}$$

$$F_{OR} = \sum_{i=1}^{l}\left(k_i + \frac{1}{\sigma_i} + \frac{1}{ub_i - lb_i}\right) * w_i \tag{10}$$

where k represents the number of projects satisfying the conditional set, k_i the number of projects satisfying only condition C_i, and σ, σ_i are the standard deviations of the effort of the k and k_i projects, respectively.

By using the standard deviation in the fitness evaluation we promote the evolved individuals that have their effort values close to the mean effort value of either the k projects satisfying S (AND case) or either the k_i projects satisfying C_i (OR case). Additionally, the evaluation rewards individuals whose difference among the lower and upper range is minimal. Finally, w_i in equations (9) and (10) is a weighting factor corresponding to the significance given by the estimator to a certain cost attribute.

The purpose of the fitness functions is to define the appropriateness of the value ranges produced within each individual according to the ISBSG dataset. More specifically, when an individual is evaluated the dataset is used to define how many records of data (a record corresponds to a project with specific values for its cost attributes and effort) lay within the ranges of values of the individual according to the conditions used and the logical operator connecting these conditions. It should be noted at this point that in the OR case the conditional set is satisfied if at least one of its conditions is satisfied, while in the AND case all conditions in S must be satisfied. Hence, k (and σ) is unique for all ranges in the AND case, while in the OR case k may have a different value for each range i. That is why the fitness functions of the two logical operators are different. The total fitness of the population in each generation is calculated as the sum of the fitness values of the individuals in P.

Once the GA terminates the best individual is used to perform effort estimation. More specifically, in the AND case we distinguish the projects that satisfy the conditional set used to train the GA, while in the OR case the projects that satisfy one or more conditions of the set. Next we find the mean effort value ($\bar{e}$) and standard deviation (σ) of those projects. If we have a new project for which we want to estimate the corresponding development effort, we first check whether the values of its attributes lay within the ranges of the best individual and that it satisfies the form of the conditional set (AND or OR). If this holds, then the effort of the new project is estimated to be:

$$e_{pred} = \bar{e} \pm \sigma \tag{11}$$

where e_{pred} is the mean value of the effort of the projects satisfying the conditional set S.

4. Experimental process

This section explains in detail the series of experiments conducted and also presents some preliminary results of the methodology. The methodology was tested on the three different datasets described in the previous section.

4.1 Design of the experiments

Each dataset was separated into two smaller sub-datasets, the first of which was used for training and the second for validation. This enables the assessment of the generalization and optimization ability of the algorithm, firstly under training conditions and secondly with new, unknown to the algorithm, data. At first, a series of initial setup experiments was performed to define and tune the parameters of the GA. These are summarized in Table 1. The values for the GA parameters were set after experimenting with different generation epochs, as well as mutation and crossover rates and various number of points of crossover. A number of control parameters were modified for experimenting and testing the sensitivity of the solution to their modification.

Category	Value	Details
Attributes set	$\{ S_{AND}, S_{OR} \}$	
Solution representation	L	
Generation size	1000 epochs	
Population size	100 individuals	
Selection		Roulette wheel based on fitness of each individual
Elitism		Best individuals are forwarded (5%)
Mutation	Ratio 0.01-0.05	Random mutation
Crossover	Ratio 0.25-0.5	Random crossover (inter-, intra-)
Termination criterion		Generations size is reached or no improvements are noted for more than 100 generations

Table 1. Genetic Algorithm main parameters

We then proceeded to produce a population of 100 individuals representing conditional sets S (or ranges of values coupled with OR or AND conditions), as opposed to the discrete values of the attributes found in the ISBSG dataset. These quantities, as shown in equations (2a) and (2b), were generated to cover a small range of values of the corresponding attributes, but are closely related to (or within) the actual values found in the original data series.

Throughout an iterative production of generations the individuals were evaluated using the fitness functions specified in equations (9) or (10) with respect to the approach adopted. As previously mentioned, this fitness was assessed based on the:

- Standard deviation

- Number of projects in *L* satisfying (7) and (8)
- Ranges produced for the attributes

Fitness is also affected by the weights given by the estimator to separate between more and less important attributes. From the fitness equations we may deduce that the combination of a high number of projects in *L*, a low standard deviation with respect to the mean effort and a small range for the cost attributes (at least the most significant) produces high fitness values. Thus, individuals satisfying these specific requirements are forwarded to the next population until the algorithm terminates. Figure 3 depicts the total fitness value of a sample population through generations, which, as expected, rises as the number of epochs increases. A plateau is observed in the range 50-400 epochs which may be attributed to a possible trapping of the GA to a local minimum. The algorithm seems to escape from this minimum with its total fitness value constantly being improved along the segment of 400-450 epochs and then stabilizing. Along the repetitions of the GA algorithm execution, the total population fitness improves showing that the methodology performs consistently well.

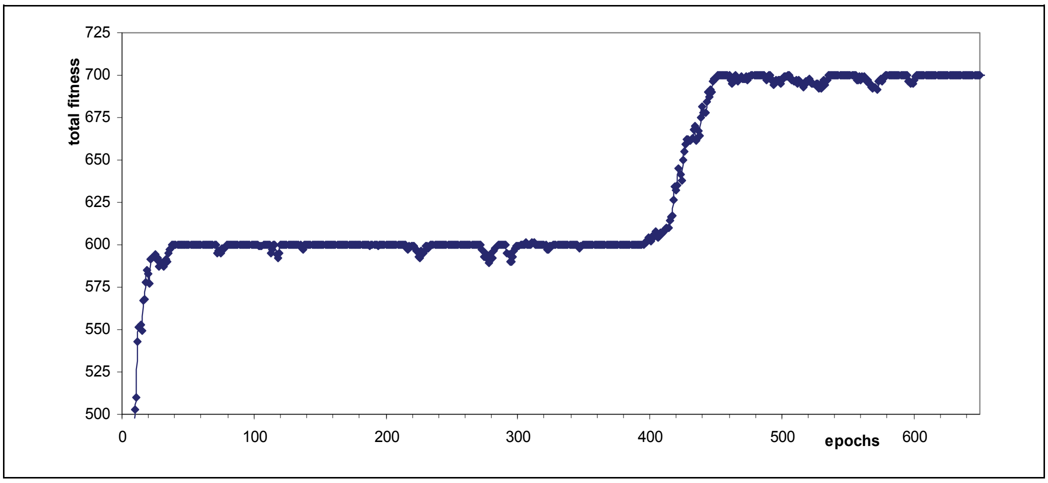


Fig. 3. Total Fitness Evolution

4.2 Experimental results

The experimental evaluation procedure was based on both the AND and OR approaches. We initially used the attributes of the datasets with equal weight values and then subsequently with combinations of different weight values. Next, as the weight values were modified it was clear that various assumptions about the importance of the given attributes for software effort could be drawn. In the first dataset for example, the Adjusted Function Point (AFP) attribute was found to have a minimal effect on development effort estimations and therefore we decided to re-run the experiments without this attribute taking part. The process was repeated for all attributes of the dataset by continuously updating the weight values and reducing the number of attributes participating in the experiments, until no more insignificant attributes remained in the dataset. The same process was followed for all the three datasets respectively, while the results summarized in this section represent only a few indicative results obtained throughout the total series of experiments.

Tables 2 and 3 present indicative best results obtained with the OR and AND approaches, respectively, that is, the best individual of each run for a given set of weights (significance) that yield the best performance with the first dataset (DS-1). Table 4 presents the best results obtained with the AND and OR approach with the second dataset (DS-2) and Table 5 lists the best obtained results with the third attribute dataset (DS-3).

Attribute Weights / Ranges				Evaluation Metrics		
FC	AC	CC	EC	ē	σ	HR
0.1 [11, 240]	0.4 [1, 1391]	0.1 [206, 1739]	0.4 [14, 169]	3014.2	1835.1	81/179
0.3 [11, 242]	0.1 [1, 1363]	0.4 [60, 1498]	0.2 [1, 350]	3125.5	1871.5	81/184
0.3 [11, 242]	**0.4 [1, 1391]**	**0.1 [1616, 2025]**	**0.2 [14, 268]**	**3204.5**	**1879.2**	**81/187**
0.2 [19, 298]	0.4 [1, 1377]	0.1 [1590, 3245]	0.3 [14, 268]	3160.3	1880.7	81/178
0.2 [11, 240]	0.2 [1, 1377]	0.4 [46, 573]	0.2 [1, 350]	3075.1	1857.2	79/183
0.2 [3, 427]	**0.4 [1, 1377]**	**0.2 [46, 579]**	**0.2 [1, 347]**	**3254.5**	**1857**	**83/191**

Table 2. Indicative Results of conditional sets using the OR approach and DS-1

Evaluation metrics were used to assess the success of the experiments, based on (i) the total mean effort, (ii) the standard deviation and, (iii) the hit ratio. The hit ratio (given in equation (12)) provides a complementary piece of information about the results. It basically assesses the success of the best individual evolved by the GA on the testing set. Recall that the GA results in conditional set of value ranges which are used to compute the mean effort and standard deviation of the projects satisfying the conditional set. Next, the number of projects n in the testing set that satisfy the conditional set is calculated. Of those n projects we compute the number of projects b that have additionally a predicted effort value satisfying equation (11). The latter may be called the "hit-projects". Thus, equation (12) essentially calculates the ratio of hit-projects in the testing set:

$$hit\ ratio(HR) = \frac{b}{n} \tag{12}$$

The results are expressed in a form satisfying equations (3)-(8). A numerical example could be a set of range values produced to satisfy equations (2a) and (2b) coupled with the logical operator of AND as follows:

$$S_{AND} = [1700, 2000] \wedge [16, 205] \wedge \ldots \wedge [180\ 200] \tag{13}$$

The projects that satisfy equation (7) are then accumulated in set L (numbers represent project IDs):

$$L=\{1827, 1986, 1987,\ldots,1806\} \tag{14}$$

Using *L* the *ē*, *σ* and HR figures may be calculated. The success of the experiments is a combination of the aforementioned metrics. Finally, we characterize an experiment as successful if its calculated standard deviation is adequately lower than the associated mean effort and achieves a hit ratio above 60%.

Indicative results of the OR conditional sets are provided in Table 2. We observe that the OR approach may be used mostly for comparative analysis of the cost attributes by evaluating their significance in the estimation process, rather the estimation itself, as results indicate low performance. Even though the acceptance level of the hit ratio is better than average, the high value of the standard deviation compared to the mean effort (measured in person days) indicates that the results attained are dispersed and not of high practical value. The total mean effort of the best 100 experiments was found equal to 2929 and the total standard deviation equal to 518. From these measures the total standard error was estimated at 4.93, which is not satisfactory, but at the same time it cannot be considered bad. However, in terms of suggesting ranges of values for specific cost attributes on which one may base an estimation, the results do not converge to a clear picture. It appears that when evaluating different groups of data in the dataset we attain large dissimilarities, suggesting that clustered groups of data may be present in the series. Nevertheless, various assumptions can be drawn from the methodology as regards to which of the attributes seem more significant and to what extent. The selected attributes, namely Added Count (AC), File Count (FC), Changed Count (CC) and Enquiry Count (EC) seem to have a descriptive role over effort as they provide results that may be considered promising for estimating effort. Additionally, the best results of Table 2 (in bold) indicate that the leading factor is Added Count (AC), with its significance being ranked very close to that of the File Count (FC).

Attribute Weights / Ranges			**Evaluation Metrics**		
FC	**AC**	**CC**	**ē**	**σ**	**HR**
0.1 [22, 223]	0.2 [187, 504]	0.7 [9, 195]	3503	1963.6	3/4
0.5 [22, 223]	0.3 [114, 420]	0.2 [9, 197]	3329.4	2014.2	3/4
0.2 [14, 156]	**0.4 [181, 489]**	**0.4 [9, 197]**	**3778.8**	**2061.4**	**3/4**
0.4 [22, 223]	**0.4 [167, 390]**	**0.2 [9, 195]**	**3850.3**	**2014.3**	**3/4**
0.2 [14, 154]	0.8 [35, 140]	0 0	2331.2	1859.4	12/16
0.7 [14, 152]	0.3 [35, 141]	0 0	2331.2	1859.4	12/16

Table 3. Indicative Results of conditional sets using the AND approach with DS-1

On the other hand, the AND approach (Table 3) provides more solid results since it is based on a more strict method (i.e. satisfy all ranges simultaneously). The results indicate again some ranking of importance for the selected attributes. To be specific, Added Count (AC) and File Count (FC) are again the dominant cost attributes, a finding which is consistent with the OR approach. We should also note that the attribute Enquiry Count (EC) proved rather insignificant in this approach, thus it was omitted from Table 3. Also, the fact that the results produced converge in terms of producing similar range bounds shows that the methodology may provide empirical indications regarding possible real attribute ranges. A high hit ratio of 75% was achieved for nearly all experiments in the AND case for the specified dataset, nevertheless this improvement is obtained with fewer projects, as expected, satisfying the strict conditional set compared to the more loose OR case. This led us to conclude that the specific attributes can provide possible ranges solving the problem and providing relatively consistent results.

The second dataset (DS-2) used for experimentation included the Normalized AFP (NAFP) and some of the previously investigated attributes for comparison purposes. The dataset was again tested using both the AND and OR approaches. The first four rows of the results shown in Table 4 individuals were obtained with the AND approach and the last two results with the OR approach. The figures listed in Table 4 show that the method 'approves' more individuals (satisfying the equations) because the ranges obtained are wider. Conclusively, the values used for effort estimation result to increase of the total standard error. The best individuals (in bold) were obtained after applying box-plots in relation to the first result shown, while the rest two results did not use this type of filtering. It is clear from the lowering of the value of the standard deviation that after box-plot filtering on the attributes some improvement was indeed achieved. Nevertheless, the HR stays quite low, thus we cannot argue that the ranges of values produced are robust to provide new effort estimates.

Attribute Weights / Ranges				Evaluation Metrics		
NAFP	**AC**	**FC**	**EC**	**$\bar{e}$**	**σ**	**HR**
0.25 [2, 134]	0.25 [215, 1071]	0.25 [513, 3678]	0.25 [4, 846]	11386.7	9005.4	9/58
0.25 [7, 80]	**0.25 [34, 830]**	**0.25 [88, 1028]**	**0.25 [37, 581]**	**2861.7**	**2515.9**	**7/66**
0.25 [1,152]	**0.25 [22, 859]**	**0.25 [58, 3192]**	**0.25 [20, 563]**	**3188.6**	**2470.9**	**3/221**
0.25 [1, 156]	**0.25 [34, 443]**	**0.25 [122, 2084]**	**0.25 [37, 469]**	**3151.6**	**2377.9**	**4/139**
0.25 [1, 36]	0.25 [449, 837]	0.25 [23, 1014]	0.25 [7, 209]	4988.5	8521.2	10/458
0.25 [1, 159]	0.25 [169, 983]	0.25 [78, 928]	0.25 [189, 567]	4988.5	8521.2	10/458

Table 4. Indicative Results of conditional sets using the AND and OR approaches with DS-2

The purpose of the final dataset (DS-3) used in the experiments is to test whether a selected subset of attributes that can be measured early in the development life-cycle can provide adequately good predictions. Results showed that the attributes of Adjusted Function Points

(AFP), Project Delivery Rate (PDRU), Project Elapsed Time (PET), Resource Level (RL) and Average Team Size (ATS) may provide improvements for selecting ranges with more accurate effort estimation abilities. For these experiments only the AND approach is presented as the results obtained were regarded to be more substantial. In the experiments conducted with this dataset (DS-3) we tried to impose even stricter ranges, after the box-plots and outlier's removal in the initial dataset, by applying an additional threshold to retain the values falling within the 90% (percentile). This was performed for the first result listed in Table 5, whereas the threshold within the 70% percentile was also applied for the second result listed on the same table. We noticed that this led to a significant optimization of the results. Even though very few individuals are approved, satisfying the equations, the HR is almost always equal to 100%. The obtained ranges are more clearly specified and in addition, sound predictions can be made regarding effort since the best obtained standard deviation of effort falls to 74.9 which also constitutes one of the best predictions yielded by the methodology. This leads us to conclude that when careful removal of outliers is performed the proposed methodology may be regarded as achieving consistently successful predictions, yielding optimum ranges that are adequately small and suggesting effort estimations that lay within reasonable mean values and perfectly acceptable deviation from the mean.

Attribute Weights / Ranges					**Evaluation Metrics**		
AFP	**PDRU**	**PET**	**RL**	**ATS**	**ē**	**σ**	**HR**
0.2 [48, 1207]	0.2 [1, 7]	0.2 [3, 12]	0.2 [1, 3]	0.2 [2, 5]	2657.6	913.7	3/3
0.2 [57, 958]	**0.2 [2, 13]**	**0.2 [5, 10]**	**0.2 [2, 4]**	**0.2 [1, 6]**	**2131.0**	**74.9**	**2/2**
0.2 [133, 1409]	0.2 [1, 25]	0.2 [7, 21]	0.2 [2, 4]	0.2 [2, 10]	2986.6	1220.0	5/5
0.2 [173, 1131]	**0.2 [1, 20]**	**0.2 [2, 20]**	**0.2 [2, 4]**	**0.2 [1, 7]**	**2380.0**	**434.5**	**2/3**
0.25 [189, 1301]	**0.25 [2, 26]**	**0 0**	**0.25 [1, 3]**	**0.25 [2, 11]**	**2477.8**	**838.2**	**5/5**
0.25 [693, 1166]	0.25 [2, 23]	0 0	0.25 [1, 3]	0.25 [1, 4]	2478.0	565.6	2/2

Table 5. Indicative Results of conditional sets using the AND approach with DS-3

5. Conclusions

In this approach we aimed at addressing the problem of large variances found in available historical data that are used in software cost estimation. Project data is expensive to collect, manage and maintain. Therefore, if we wish to lower the dependence of the estimation to the need of gathering accurate and homogenous data, we might consider simulating or generating data ranges instead of real crisp values.

The theory of conditional sets was applied in the present work with Genetic Algorithms (GAs) on empirical software cost estimation data. GAs are ideal for providing efficient and effective solutions in complex problems; there are, however, several trade-offs. One of the major difficulties in adopting such an approach is that it requires a thorough calibration of the algorithm's parameters. We have tried to investigate the relationship between software attributes and effort, by evolving attribute value ranges and evaluating estimated efforts. The algorithm promotes the best individuals in the reproduced generations through a probabilistic manner. Our methodology attempted to reduce the variations in performance of the model and achieve some stability in the results. To do so we approached the problem from the perspective of minimizing the differences in the ranges and the actual and estimated effort values to decisively determine which attributes are the most important in software cost estimates.

We used the ISBSG repository containing a relatively large quantity of data; nevertheless, this data suffers from heterogeneity thus presents low quality level from the perspective of level of values. We formed three different subsets selecting specific cost attributes from the ISBSG repository and filtering out outliers using box-plots on these attributes. Even though the results are of average performance when using the first two datasets, they indicated some importance ranking for the attributes investigated. According to this ranking, the attributes Added Count (AC) and File Count (FC) were found to lay among the most significant cost drivers for the ISBSG dataset. The third dataset included Adjusted Function Points (AFP), Project Delivery Rate (PDRU), Project Elapsed Time (PET), Resource Level (RL) and Average Team Size (ATS). These attributes may be measured early in the software life-cycle, thus this dataset may be regarded more significant than the previous two from a practical perspective. A careful and stricter filtering of this dataset provided prediction improvements, with the yielded results suggesting small value ranges and fair estimates for the mean effort of a new project and its deviation. There was also an indication that within different areas of the data, significantly different results may be produced. This is highly related to the scarcity of the dataset itself and supports the hypothesis that if we perform some sort of clustering in the dataset we may further minimize the deviation differences in the results and obtain better effort estimates.

Although the results of this work are at a preliminary stage it became evident that the approach is promising. Therefore, future research steps will concentrate on ways to improve performance, examples of which may be: (i) Pre-processing of the ISBSG dataset and appropriate clustering into groups of projects that will share similar value characteristics. (ii) Investigation of the possibility of reducing the attributes in the dataset by utilizing a significance ranking mechanism that will promote only the dominant cost drivers. (iii) Better tuning of the GA's parameters and modification/enhancement of the fitness functions to yield better convergence. (iv) Optimization of the trial and error weight factor assignment used in the present work by utilizing a GA. (v) Experimentation with other datasets containing selected attributes again proposed by a GA. Finally, we plan to perform a comparative evaluation of the proposed approach with other well established algorithms, like for example the COCOMO model.

6. References

Adamopoulos, A.V.; Likothanassis, S.D. & Georgopoulos, E.F. (1998). A Feature Extractor of Seismic Data Using Genetic Algorithms, *Signal Processing IX: Theories and Applications, Proceedings of EUSIPCO-98, the 9th European Signal Processing Conference*, Vol. 2, pp. 2429-2432, Typorama, Greece.

Albrecht, A.J. & Gaffney J.R. (1983). Software Function Source Lines of Code, and Development Effort Prediction: A Software Science Validation, *IEEE Transactions on Software Engineering*. Vol. 9, No. 6, pp. 639-648.

Boehm, B.W. (1981). *Software Engineering Economics*, Prentice Hall, New Jersey.

Boehm, B.W.; Clark, B.; Horowitz, E.; Westland, C.; Madachy, R.J. & Selby R.W. (1995). Cost Models for Future Software Life Cycle Processes: COCOMO 2.0, *Annals of Software Engineering*, Vol.1, pp. 57-94, Springer, Netherlands.

Boehm, B.W.; Abts, C. & Chulani, S. (2000). Software Development Cost Estimation Approaches – A Survey, *Annals of Software Engineering*, Vol.10, No. 1, pp. 177-205, Springer, Netherlands.

Burgess, C.J. & Lefley, M. (2001). Can Genetic Programming Improve Software Effort Estimation? A Comparative Evaluation, *Information and Software Technology*, Vol. 43, No. 14, pp. 863-873, Elsevier, Amsterdam.

Dolado, J.J. (2000). A Validation of the Component-Based Method for Software Size Estimation, *IEEE Transactions on Software Engineering*, Vol. 26, No. 10, pp. 1006-1021, IEEE Computer Press, Washington D.C..

Dolado, J.J. (2001). On the Problem of the Software Cost Function, *Information and Software Technology*, Vol. 43, No. 1, pp. 61-72, Elsevier, Amsterdam.

Fairley, R.E. (1992). Recent Advances in Software Estimation Techniques, *Proceedings of the 14th International Conference on Software Engineering*, pp. 382-391, ACM, Melbourne, Australia.

Finnie, G.R.; Wittig, G.E. & Desharnais, J.-M. (1997). Estimating software development effort with case-based reasoning, *Proceedings of the 2nd International Conference on Case-Based Reasoning Research and Development ICCBR*, pp.13-22, Springer.

Holland, J.H. (1992). Genetic Algorithms, *Scientific American*, Vol. 267, No. 1, pp. 66–72, New York.

Huang, S. & Chiu, N. (2006). Optimization of analogy weights by genetic algorithm for software effort estimation, *Information and Software Technology*, Vol. 48, pp. 1034-1045, Elsevier.

Idri, A.; Khoshgoftaar, T.M. & Abran, A. (2002). Can Neural Networks be Easily Interpreted in Software Cost Estimation?, *Proceedings of the 2002 IEEE World Congress on Computational Intelligence*, pp. 1162-1167 IEEE Computer Press, Washington D.C..

International Software Benchmarking Standards Group (ISBSG), Estimating, Benchmarking & Research Suite Release 9, ISBSG, Victoria, 2005.

International Software Benchmarking Standards Group, http://www.isbsg.org/

Jorgensen, M. & Shepperd, M. (2007). A Systematic Review of Software Development Cost Estimation Studies, *IEEE Transactions on Software Engineering*, Vol. 33, No. 1, pp. 33-53, IEEE Computer Press, Washington D.C..

Jun, E.S. & Lee, J.K. (2001). Quasi-optimal Case-selective Neural Network Model for Software Effort Estimation, *Expert Systems with Applications*, Vol. 21, No. 1, pp. 1-14 Elsevier, New York.

Khoshgoftaar, T.M.; Evett, M.P.; Allen, E.B. & Chien, P. (1998). An Application of Genetic Programming to Software Quality Prediction *Computational Intelligence in Software Engineering, Series on Advances in Fuzzy Systems – Applications and Theory*, Vol. 16, pp. 176-195, World Scientific, Singapore.

Koza, J.R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Massachusetts.

Lederer, A.L. & Prasad, J. (1992). Nine Management Guidelines for Better Cost Estimating, *Communications of the ACM*, Vol. 35, No. 2, pp. 51-59, ACM, New York.

Lefley, M. & Shepperd, M.J. (2003). Using Genetic Programming to Improve Software Effort Estimation Based on General Data Sets, *Proceedings of GECCO*, pp. 2477-2487.

MacDonell, S.G. & Shepperd, M.J. (2003). Combining Techniques to Optimize Effort Predictions in Software Project Management, *Journal of Systems and Software*, Vol. 66, No. 2, pp. 91-98, Elsevier, Amsterdam.

Mair, C; Kadoda, G.; Lefley, M.; Phalp, K.; Schofield, C.; Shepperd, M. & Webster, S. (2000). An investigation of machine learning based prediction systems, *Journal of Systems Software*, Vol. 53, pp. 23–29, Elsevier.

Meyer, T.P. & Packard, N.H. (1992). Local Forecasting of High-dimensional Chaotic Dynamics, *Nonlinear Modeling and Forecasting*, Addison-Wesley.

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin.

Packard, N.H. (1990). A Genetic Learning Algorithm for the Analysis of Complex Data, *Complex Systems*, Vol. 4, No. 5, pp. 543-572, Illinois.

Pendharkar, P.C.; Subramanian, G.H. & Rodger, J.A. (2005). A Probabilistic Model for Predicting Software Development Effort, *IEEE Transactions on Software Engineering*, IEEE Computer Press, Vol. 31, No. 7, pp. 615-624, Washington D.C..

Papatheocharous, E. & Andreou, A. (2007). Software Cost Estimation Using Artificial Neural Networks with Inputs Selection, *Proceedings of the 9th International Conference on Enterprise Information Systems*, pp. 398-407, Madeira, Portugal.

Putnam, L.H. & Myers, W. (1992). *Measures for Excellence, Reliable Software on Time, Within Budget*, Yourdan Press, New Jersey.

Shepperd, M. & Kadoda, G. (2001). Comparing Software Prediction Techniques Using Simulation, *IEEE Transactions on Software Engineering*, Vol. 27, No. 11, pp. 1014-1022, IEEE Computer Press, Washington D.C..

Shepperd, M.J.; Schofield, C. & Kitchenham, B. A. (1996). Effort estimation using analogy, *Proceedings of the 18th International Conference on Software Engineering*, pp. 170-178, Berlin.

Shukla, K.K. (2000). Neuro-genetic Prediction of Software Development Effort, *Information and Software Technology*, Vol. 42, No. 10, pp. 701-713, Elsevier, Amsterdam.

Tadayon, N. (2005). Neural Network Approach for Software Cost Estimation. *Proceedings of the International Conference on Information Technology: Coding and Computing*, pp. 815-818, IEEE Computer Press, Washington D.C..

Xu, Z. & Khoshgoftaar, T.M. (2004). Identification of Fuzzy Models of Software Cost Estimation, *Fuzzy Sets and Systems*, Vol. 145, No. 1, pp. 141-163, Elsevier, New York.

2

Towards Intelligible Query Processing in Relevance Feedback-Based Image Retrieval Systems

Belkhatir Mohammed
Faculty of IT, Monash University
???

1. Introduction

We propose in this paper the specification of an image retrieval architecture based on a relevance feedback framework which operates on high-level image descriptions instead of their extracted low-level features. This framework features a conceptual model which integrates visual semantics as well as symbolic relational characterizations and operates on image objects, abstractions of visual entities within a physical image. Also, it manipulates a rich query language, consisting of both boolean and quantification operators, which therefore leads to optimized user interaction and increased retrieval performance. Let us first introduce the context of our research.

In order to cope with the storing and retrieval of ever-growing digital image collections, the first retrieval systems (cf. [Smeulders et al. 00] for a review of the state-of-the-art), known as content-based, propose fully automatic processing methods based on low-level signal features (color, texture, shape...). Although they allow the fast processing of queries, they do not make it possible to search for images based on their semantic content and consider for example red apples or Ferraris as being the same entities simply because they have the same color distribution. Failing to relate low-level features to semantic characterization (also known as the **semantic gap)** has slowed down the development of such solutions since, as shown in [Hollink 04], taking into account aspects related to the image content is of prime importance for efficient retrieval. Also, users are more skilled in defining their information needs using language-based descriptors and would therefore rather be given the possibility to differentiate between red roses and red cars.

In order to overcome the semantic gap, a class of frameworks within the framework of the European Fermi project proposed to model the image semantic and signal contents following a sharp process of human-assisted indexing [Mechkour 95] [Meghini et al. 01]. These approaches, based on elaborate knowledge-based representation models, provide satisfactory results in terms of retrieval quality but are not easily usable on large collections of images because of the necessary human intervention required for indexing.

Automated systems which attempt to deal with the semantics/signal integration (e.g. iFind [Lu et al. 00] and the prototype presented in [Zhou & Huang 02]) propose solutions based on textual annotations to characterize semantics and on a relevance feedback (RF) scheme operating on low-level features. RF techniques are based on an interaction with a user providing judgment on displayed images as to whether and to what extent they are relevant or irrelevant to his need. For each loop of the interaction, these images are learnt and the system tries to display images close in similarity to the ones targeted by the user. As any learning process, it requires an important number of training images to achieve reasonable performance. The user is therefore solicited through several tedious and time-consuming loops to provide feedback for the system in real time, which penalizes user interaction and involves costly computations over the whole set of images. Moreover, starting from a textual query on semantics, these state-of-the art systems are only able to manage **opaque** RF (i.e. a user selects relevant and/or non-relevant documents and is then proposed a revised ranking without being given the possibility to 'understand' how his initial query was transformed) since it operates on extracted low-level features. Finally, these systems do not take into account the relational spatial information between visual entities, which affects the quality of the retrieval results.

Our RF process is a specific case of state-of-the-art RF frameworks reducing the user's burden since it involves a unique loop returning the relevant images. Moreover, as opposed to the opacity of state-of-the-art RF frameworks, it holds the advantage of being **transparent** (i.e. the system displays the query generated from the selected documents) and **penetrable** (i.e. the modification of the generated query is possible before processing), which increases the quality of retrieval results. Through the use of a symbolic representation, the user is indeed able to visualize and comprehend the intelligible query being processed. We manage transparent and penetrable interactions by considering a conceptual representation of images and model their conveyed visual semantics and relational information through a high-level and expressive representation formalism. Given a user's feedback (i.e. judgment or relevance or irrelevance), our RF process, operating on both visual semantics and relational spatial characterization, is therefore able to first generate and then display a query for eventual further modifications operated by the user. It enforces computational efficiency by generating a symbolic query instead of dealing with costly learning algorithms and optimizes user interaction by displaying this 'readable' symbolic query instead of operating on hidden low-level features.

As opposed to state-of-the-art loosely-coupled solutions penalizing user interaction and retrieval performance with an opaque RF framework operating on low-level features, our architecture combines a keyword-based module with a transparent and penetrable RF process which refines the retrieval results of the first. Moreover, we offer a rich query language consisting of several Boolean operators.

At the core of our work is the notion of **image objects** (**IOs**), abstract structures representing visual entities within an image. Their specification is an attempt to operate beyond simple low-level signal features since IOs convey the semantic and relational information.

In the remainder, we first detail the processes allowing to abstract the extracted low-level features to high-level relational description in section 2. Section 3 deals with the visual semantic characterization. We specify in section 4 the image model and develop its conceptual instantiation integrating visual semantics and relational (spatial) features. Section 5 is dedicated to the presentation of the RF framework.

2. From low-level spatial features to high-level relational description

Taking into account spatial relations between semantically-defined visual entities is crucial in the framework of an image retrieval system since it enriches the index structures and expands the query language. Also, dealing with relational information between image components allows to enhance the quality of the results of an information retrieval system [Ounis&Pasca 98]. However, relating low-level spatial characterizations to high-level textual descriptions is not a straightforward task as it involves highligting a spatial vocabulary and specifying automatic processes for this mapping. We first study in this section methods used to represent spatial data and deal with the automatic generation of high-level spatial relations following a first process of low-level extraction.

2.1 Defining a spatial vocabulary through the relation-oriented approach

We consider two types of spatial characterizations: the first describes the absolute positions of visual entities and the second their relative locations.

In order to model the spatial data, we consider the «relation-oriented» approach which allows explicitly representing the relevant spatial relations between IOs without taking into account their basic geometrical features. Our study features the four modeling and representation spaces:

- The Euclidean space gathers the image pixels coordinates. Starting with this information, all knowledge related to the other representation spaces can be inferred.
- The Topological space is itself linked to the notions of continuity and connection. We consider five topological relations and justify this choice by the fact that these relations are exhaustive and relevant in the framework of an image indexing and retrieval system. Let io1 and io2 two IOs. These relations are (s_1=**P**,io1,io2) : 'io1 is a part of io2', (s_2=**T**,io1,io2) : 'io1 touches io2 (is externally connected)', (s_3=**D**,io1,io2) : 'io1 is disconnected from io2', (s_4=**C**,io1,io2) : 'io1 partially covers (in front of) io2' and (s_5=**C_B**,io1,io2) : 'io1 is covered by (behind) io2'. Let us note that these relations are mutually exclusive and characterized by the important property that each pair of IOs is linked by only one of these relations.
- The Vectorial space gathers the directional relations: Right (s_6=**R**), Left (s_7=**L**), Above (s_8=**A**) and Below (s_9=**B**). These relations are invariant to basic geometrical transformations such as translation and scaling.
- In the metric space, we consider the fuzzy distance relations Near (s_{10}=**N**) and Far (s_{11}=**F**). Discrete relations are not considered since providing a query language which allows a user to quantify the distance between two visual entities would penalize the fluidity of the interaction.

2.2 Automatic spatial characterization

Topological relations. In our spatial modeling, an IO *io* is characterized by its center of gravity io_c and by two pixel sets: its interior, noted io_i and its border io_b. We define for an image an orthonormal axis with its origin being the image left superior border and the basic measure unity, the pixel. All spatial characterizations of an object such as its border, interior and center of gravity are defined with respect to this axis.

In order to highlight topological relations between IOs, we consider the intersections of their interior and border pixel sets through a process adapted from [Egenhofer 91]. Let io1 and io2 be two IOs, the four intersections are: io1_i ∩ io2_i, io1_i ∩ io2_b, io1_b ∩ io2_i and io1_b ∩ io2_b. Each topological relation is linked to the results of these intersections as illustrated in table 1. The strength of this computation method relies on associating topological relations to a set of necessary and sufficient conditions linked to spatial attributes of IOs (i.e. their interior and border pixel sets).

Intersections / Topological Relation	io1_b ∩ io2_b	io1_i ∩ io2_b	io1_b ∩ io2_i	io1_i ∩ io2_i
(P, io1, io2)	$\varnothing$	$\neq \varnothing$	$\varnothing$	$\neq \varnothing$
(T, io1, io2)	$\neq \varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
(D, io1, io2)	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
(C, io1, io2)	$\varnothing$	$\varnothing$	$\neq \varnothing$	$\neq \varnothing$
(C_B, io1, io2)	$\varnothing$	$\neq \varnothing$	$\varnothing$	$\neq \varnothing$

Table 1. Characterization of topological relations with the intersections of interior and border pixel sets of two IOs

Directional relations. The computation of directional relations between io1 and io2 is based on their centers of gravity io1_c($x1_c$, $y1_c$) and io2_c($x2_c$, $y2_c$), the minimal and maximal coordinates along x axis ($x1_{min}$, $x2_{min}$ & $x1_{max}$, $x2_{max}$) as well as the minimal and maximal coordinates along y axis ($y1_{min}$, $y2_{min}$ & $y1_{max}$, $y2_{max}$) of their four extremities.
We will say that io1 is at the left of io2, noted (L,io1,io2) iff. $(x1_c<x2_c) \wedge (x1_{min}<x2_{min}) \wedge (x1_{max}<x2_{max})$.
io1 is at the right of io2, noted (R,io1,io2) iff. $(x1_c>x2_c) \wedge (x1_{min}>x2_{min}) \wedge (x1_{max}>x2_{max})$.
We will say that io1 is above io2, noted (A,io1,io2) iff. $(y1_c>y2_c) \wedge (y1_{min}>y2_{min}) \wedge (y1_{max}>y2_{max})$.
io1 is below io2, noted (B,io1,io2) iff. $(y1_c<y2_c) \wedge (y1_{min}<y2_{min}) \wedge (y1_{max}<y2_{max})$.
We illustrate these definitions in figure 1 where the IO corresponding to huts (io1) is above the IO corresponding to the grass (io2). It is however not at the left of the latter since $x1_c<x2_c$ but $x1_{min}>x2_{min}$.

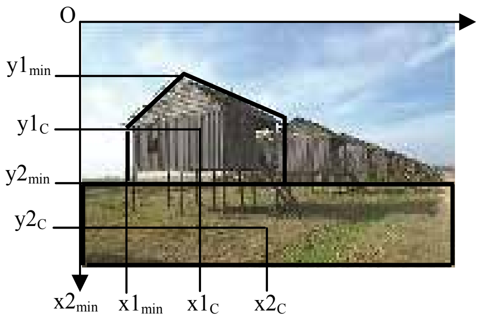


Figure 1. Characterization of directional relations

Metric relations. In order to distinguish between the Near and Far relations, we use the constant $D_{sp}= d(\vec{0},0.5*[\sigma_1,\sigma_2]^T)$ where d is the Euclidean distance between the null vector $\vec{0}$ and $[\sigma_1,\sigma_2]^T$ is the vector of standard deviations of the localization of centers of gravity for each IO in each dimension from the overall spatial distribution of all IOs in the corpus. D_{sp} is therefore a measure of the spread of the distribution of centers of gravity of IOs. This distance agrees with results from psychophysics and can be interpreted as the bigger the spread, the larger the distances between centers of gravity are. We will say that two IOs are **near** if the Euclidean distance between their centers of gravity is inferior to D_{sp}, **far** otherwise.

2.3 From low-level features to symbolic spatial relations

So as to deduct knowledge from partial spatial information and to enforce computational efficiency, composition rules are used to infer relations between two IOs io1 and io2 from the relations generated between io1, io2 and a third IO io3. For example, if io1 is at the left of io3 and io3 at the left of io2 then io1 is at the left of io2.
Composition rules on spatial relations are dynamically processed when constructing index spatial representations. Let us note moreover that there are existing implications between spatial relations characterized in different modeling spaces. We identified the following implications related to the topological relations only:

- (P,io1,io2)→ ¬ (T, io1, io2) ∧ ¬ (D, io1, io2) ∧ ¬ (C, io1, io2) ∧ ¬ (C_B, io1, io2)
- (T,io1,io2)→ ¬ (P, io1, io2) ∧ ¬ (D, io1, io2) ∧ ¬ (C, io1, io2) ∧ ¬ (C_B, io1, io2)
- (D,io1,io2)→ ¬ (P, io1, io2) ∧ ¬ (T, io1, io2) ∧ ¬ (C, io1, io2) ∧ ¬ (C_B, io1, io2)
- (C,io1,io2)→ ¬ (P, io1, io2) ∧ ¬ (T, io1, io2) ∧ ¬ (D, io1, io2) ∧ ¬ (C_B, io1, io2)
- (C_B,io1,io2)→ ¬ (P, io1, io2) ∧ ¬ (T, io1, io2) ∧ ¬ (D, io1, io2) ∧ ¬ (C, io1, io2)

These implications illustrate the fact that there exists a unique topological relation between two IOs.
We identified the following implications related to the directional relations:

- (L,io1,io2) → ¬ (R,io1,io2); (R,io1,io2) → ¬ (L,io1,io2)
- (A,io1,io2) → ¬ (B,io1,io2); (B,io1,io2) → ¬ (A,io1,io2)

These implications illustrate the fact that an IO io1 is either at the left or at the right of a second IO io2. Also, it is either above, either below io2.
We identified the following implications between metric relations only:

- (N,io1,io2) → ¬ (F,io1,io2); (F,io1,io2) → ¬ (N,io1,io2)

These implications illustrate the fact that an IO io1 is either near, either far from a second IO io2.
Finally, we identified the following implications between spatial relations of distinct natures:

- (P, io1, io2) → N, io1, io2), if io1 is **part of** io2, then it is **near** io2.
- (T, io1, io2) → (N, io1, io2), if io1 **touches** io2, then it is **near** io2.

We propose in the next section to highlight the image visual semantics, i.e. semantic concepts linked to IOs.

3. Characterizing the visual semantics

Semantic concepts are learned and then automatically extracted given a visual ontology. Its specification is strongly constrained by the application domain. Indeed, the development of cross-domain multimedia ontologies is currently limited by the difficulty to automatically map low-level signal features to semantic concepts [Naphade et al. 06]. Our efforts have been focused towards developing an ontology for general-purpose photography.
Several experimental studies presented in [Mojsilovic&Rogowitz 01] have led to the specification of twenty categories or picture scenes describing the image content at a global level. Web-based image search engines (google, altavista) are queried by textual keywords

corresponding to these picture scenes and 100 images are gathered for each query. These images are used to establish a list of semantic concepts characterizing objects that can be encountered in these scenes. A total of 72 semantic concepts to be learnt and automatically extracted are specified.

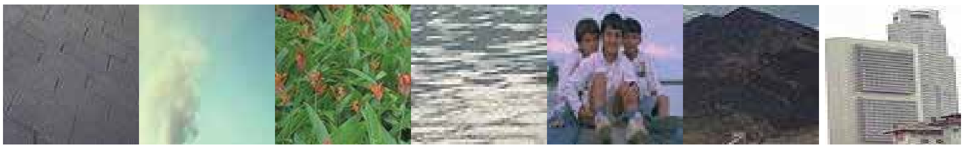
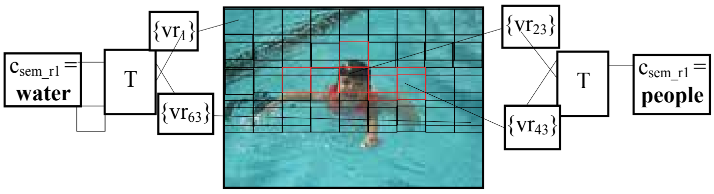
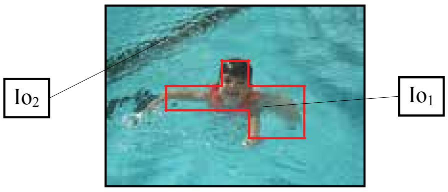


Figure 2. Image patches corresponding to semantic concepts: ground, sky, vegetation, water, people, mountain, building

A three-layer feed-forward neural network with dynamic node creation capabilities is used to learn these semantic concepts. Labeled image patches cropped from home photographs constitute the training corpus T (example images are provided in figure 3). Low-level color and texture features are computed for each of the training images as an input vector for the neural network.



a)Learning framework linking each grid-based region with a semantic-concept and its recognition result



b)Recognition results are reconciled across all regions to highlight IOs

Figure 3. Architecture for the highlighting of IOs and the characterization of their corresponding semantic concept

Once the neural network has learned the visual vocabulary, the approach subjects an image to be indexed to a multi-scale, grid-based recognition against these semantic concepts. An image to be processed is scanned with grids of several scales. Each one features visual regions $\{vr_i\}$ characterized by a feature vector of low-level color and texture features. The latter is compared against feature vectors of labeled image patches corresponding to semantic concepts in the training corpus T (figure 3.a)). Recognition results for all semantic concepts are computed and then reconciled across all grid regions which are aggregated according to configurable spatial tessellation (figure 3.b)) in order to highlight IOs. Each IO is linked to a semantic concept with maximum recognition value.

4. A model for semantic/relational integration

We propose an image model combining visual semantics and relational characterization through a bi-facetted representation (cf. figure 4). The image model consists of both a physical image level representing an image as a matrix of pixels and a conceptual level. IOs convey the visual semantics and the relational information at the conceptual level. The latter is itself a bi-facetted framework:

- The **visual semantics facet** describes the image semantic content and is based on labeling IOs with a semantic concept. E.g., in figure 4, the second IO (Io2) is tagged by the semantic concept *Water*. Its conceptual specification is dealt with in section 4.1.
- The **relational facet** features the image relational content in terms of symbolic spatial relations. E.g., in figure 4, Io1 is **inside** Io2. Its conceptual specification is dealt with in section 4.2.

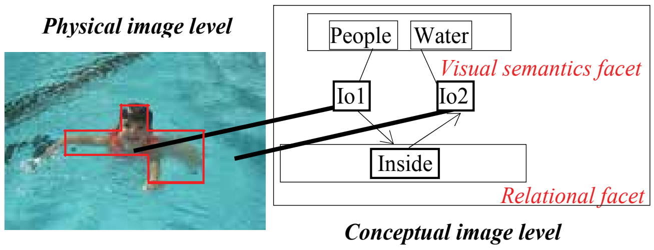


Figure 4. Image Model

To instantiate this model within an image retrieval framework, we use a representation formalism capable to model IOs as well as the conveyed visual semantics and relational information. This formalism should moreover make it easy to visualize the image information, especially as far as the interaction with the user within a RF framework is concerned. A graph-based representation and particularly conceptual graphs (CGs) [Sowa 84] is an efficient solution to describe an image and characterize its components. CGs have indeed proven to adapt to the symbolic approach of image retrieval [Mechkour 96] [Belkhatir et al. 04] [Belkhatir 05a] [Belkhatir et al. 05b]. CGs allow to represent components of our image retrieval architecture and to specify expressive index and query frameworks. Formally, a CG is a finite, bipartite and directed graph. It features two types of nodes: concept and relation nodes. In the graph [Tools with Artificial Intelligence]←(Entitled) ←[Book]→(Published_by)→[I-Tech], concepts are between brackets and relations between parentheses. This graph is equivalent to a first-order logical expression where concepts and relations are connected by the conjunction operator (boolean **AND**):
$\exists$ x,y,z s.t. **(**Book=x) $\wedge$ (Tools with Artificial Intelligence=y) $\wedge$ (I-Tech=z) $\wedge$ Entitled(x,y) $\wedge$ Published_by(x,z).
It is semantically interpreted as: the book entitled Tools with Artificial Intelligence is published by I-Tech. Concepts and conceptual relations are organized within a lattice structure partially ordered by the IS-A ($\leq$) relation. Person $\leq$ Man, e.g., denotes that the concept *Man* is a specialization of the concept *Person*, and will therefore appear in the offspring of the latter within the lattice organizing these concepts. In our model, CGs are used to represent the image content at the conceptual level.

4.1 Representation of the visual semantics facet

An instance of the visual semantics facet is represented by a set of CGs, each one containing an *Io* concept linked through the conceptual relation *is_a* to a semantic concept: [Io]→(is_a)→[c_{sem}[i]]. E.g., graphs [Io1]→(is_a)→[People] and [Io2]→(is_a)→[Water] are the representation of the visual semantics facet in figure 4 and can be translated as: the first IO (Io1) is associated with the semantic concept *people* and the second IO (Io2) with the semantic concept *water*. We use WordNet to elaborate a visual ontology that reflects the is_a relation among the semantic concepts. They are organized within a multi-layered lattice ordered by a specific/generic partial order (a part of the lattice is given in figure 5).

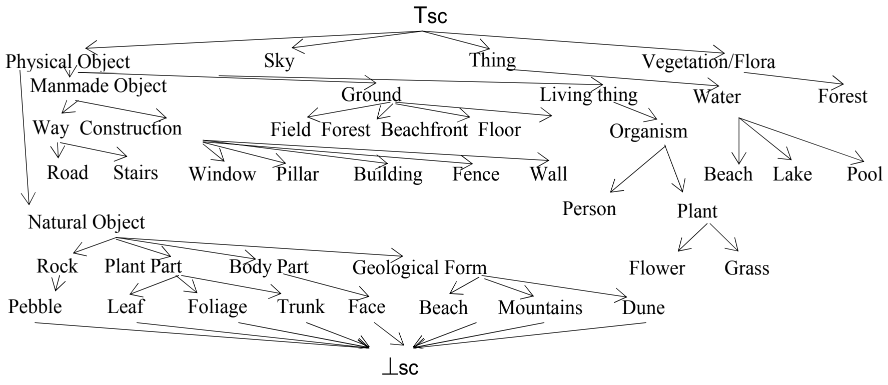


Figure 5. Lattice organizing semantic concepts

We now focus on the relational facet by first proposing structures for the integration of relational information within our strongly-integrated framework and then specifying their representation in terms of CGs.

4.2 Conceptual representation of the relational facet

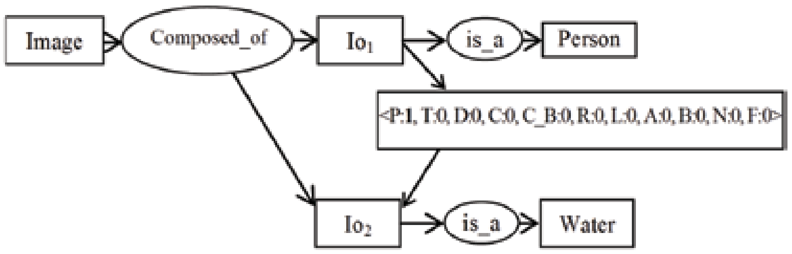
Each pair of IOs are related through an index spatial meta-relation (ISR), compact structure summarizing spatial relationships between these IOs. ISRs are supported by a vector structure **Sp** with eleven elements corresponding to the previously explicited spatial relations. Values Sp[i], $i \in [1,11]$ are booleans stressing that the spatial relation s_i links the two considered IOs. E.g., the first and second IOs (Io2) respectively corresponding to semantic concepts *person* and *water* in figure 4 are related by the ISR <P:**1**, T:0, D:0, C:0, C_B:0, R:0, L:0, A:0, B:0, N:0, F:0>, which is translated by Io1 being **inside** (**part of**) Io2.

Our framework proposes an expressive query language which integrates visual semantics and symbolic spatial characterization through boolean operators. A query which associates visual semantics with a boolean disjunction of spatial relations such as **Q**: "Find images with people **at the left** OR **at the right** of buildings" can therefore be processed (user-formulated queries are studied in [Belkhatir 05b]). *Or* spatial concepts (OSCs) are conceptual structures semantically linked to the disjunction boolean operator and specified for the processing of such a query. They are supported by the vector structure $\mathbf{Sp_{or}}$ such that $Sp_{or}(i)$, $i \in [1,11]$, is a non-null boolean value if the spatial relation s_i is mentioned in the disjunction of spatial relations within the query. The OSR <P:0, T:0, D:0, C:0, C_B:0, R:**1**, L:**1**, A:0, B:0, N:0, F:0>$_{OR}$ corresponds to the spatial characterization expressed in Q.

In our conceptual representation of the spatial facet, spatial meta-relations are elements of partially-ordered lattices organized with respect to the type of the query processed. There are two types of basic graphs controlling the generation of all the relational facet graphs. **Index spatial graphs** link two IOs through an ISR: **[Io1]→(ISR)→[Io2]**. **Query spatial graphs** link two IOs through *And*, *Or* or *Not* spatial meta-relations **[Io1]→(ASR)→[Io2]**; **[Io1]→(OSR)→[Io2]** and **[Io1]→(NSR)→[Io2]**. Eg, the index spatial graph [Io1]→(<P:**1**, T:0, D:0, C:0, C_B:0, R:0, L:0, A:0, B:0, N:0, F:0>)→[Io2] is the index representation of the spatial facet in figure 4 and is interpreted as the first IO (Io1) is related to the second IO (Io2) through the ISR <P:**1**, T:0, D:0, C:0, C_B:0, R:0, L:0, A:0, B:0, N:0, F:0>. The query spatial graph [Io1]→(<P:0, T:0, D:0, C:0, C_B:0, R:**1**, L:**1**, A:0, B:0, N:0, F:0>$_{OR}$)→[Io2] is the representation of query Q.

4.3 Image index and query representations

Image index and query representations are obtained through the combination (**join** operation [Sowa 84]) of CGs over the visual semantics and relational facets. We propose the graph unifying all visual semantics and spatial CG representations of the image proposed in figure 4:



5. A relevance feedback framework strongly integrating visual semantics and relational descriptions

We present a RF framework enhancing the state-of-the-art techniques as far as two major issues are concerned. First, while most image RF architectures are designed to deal with global image features, our framework operates at the IO level and the user is therefore able to select visual entities of interest to refine his search. Moreover, the user has a total control of the query process since the system displays the query generated from the images he selects and allows its modification before processing.

5.1 Use case scenario

Our RF framework operates on the whole corpus or on a subset of images displayed after an initial query image was proposed. The user refines his search by selecting IOs of interest. In case the user wants to refine the spatial characterization between a pair of visual entities (e.g. the user is interested in retrieving people either inside, in front of or at the right of a water area), he first queries with the semantic concepts corresponding to these entities (here '*water* and *people*') and then enrich his characterization through RF. The system translates the phrase query 'water and people' in a visual semantics graph:

$$\begin{tikzcd}[Image] \arrow[r] & (composed\_of) \arrow[r] \arrow[dr] & {[Io_1]} \arrow[r] & (is\_a) \arrow[r] & {[water]} \\ & & {[Io_2]} \arrow[r] & (is\_a) \arrow[r] & {[people]}\end{tikzcd}$$

The latter is processed and the results are given in figure 6.

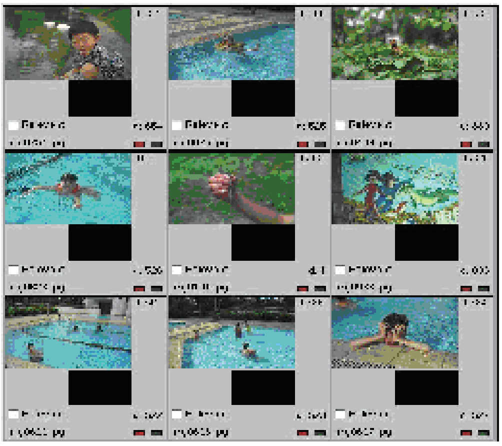


Figure 6. First retrieval for the query "water and people"

When the RF mode is chosen, the system displays all IOs within images relevant to the query 'water and people'. The user chooses to highlight 3 pairs of IOs (figure 7) within displayed images which are relevant to his need (i.e. present the specific visual semantic and spatial characterizations he is interested in).

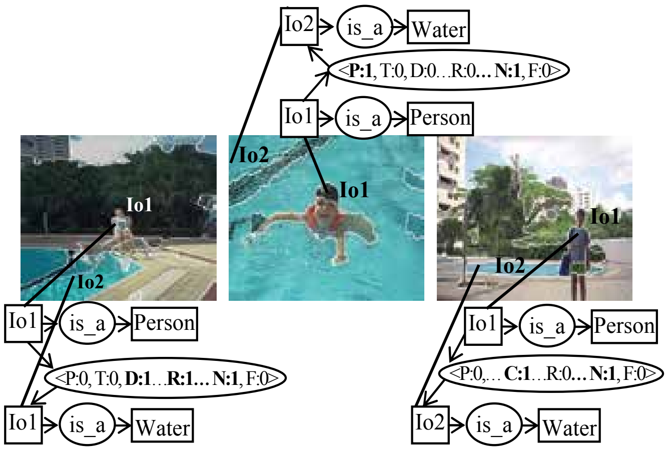
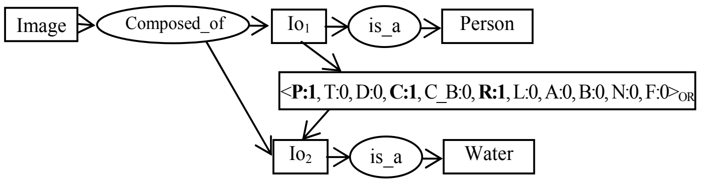

Io2
is_a
Water
<**P:1**, T:0, D:0...R:0...**N:1**, F:0>
Io1
is_a
Person
Io2
Io1
Io1
Io2
Io1
Io2
Io1
is_a
Person
<P:0, T:0, **D:1**...**R:1**...**N:1**, F:0>
Io1
is_a
Water
Io1
is_a
Person
<P:0,... **C:1**...R:0...**N:1**, F:0>
Io2
is_a
Water

Figure 7. Selected IOs and their conceptual representation

The system is then expected to generate a generalized and accurate representation of the user's need from the conceptual information conveyed by the selected IOs.
According to the user's selection, the system should find out that the user focuses on images containing a person either being inside, in front of <u>or</u> at the right of water. Our RF framework therefore processes the ISRs of the selected pairs of IOs so as to construct the OSR <**P:1**, T:0, D:0, **C:1**, C_B:0, **R:1**, L:0, A:0, B:0, N:0, F:0>$_{OR}$. The spatial query graph [Io1]→[<**P:1**, T:0, D:0, **C:1**, C_B:0, **R:1**, L:0, A:0, B:0, N:0, F:0>$_{OR}$]→[Io2] is then generated. Finally, visual semantics and spatial query graphs are aggregated to build the full query graph:



5.2 Relevance feedback algorithm

The algorithm summarizing the RF mode is as follows:

Given a query with semantic concepts SC_i, **generate** a visual semantics graph G_{sem}.
Process the graph and **display** relevant images.
If the user selects the RF mode, **highlight** IOs then **take into account** the n pairs of IOs selected by the user.
Regarding the spatial subfacet
The n selected pairs of IOs are characterized by n ISRs supported by vector structures $[Sp]_k$ ($k \in [1,n]$) such that values $[Sp(i)]_k$, $i \in [1,11]$ are booleans stressing that in the kth ISR the spatial relation s_i links the considered pair of IOs.
Generate four *Or* spatial relations respectively corresponding to the topological relations, the right/left and above/below directional relations and finally the metric relations considering the n ISRs (let us note that we generate an OSR for each group of relations which are said incompatible, i.e. one IO cannot be both at the left and at the right of an other IO, also one IO cannot be both near and far from an other IO etc…). These OSRs are supported by vector structures $[Sp_{OR}]_j(i)$, $j \in [1,4]$, $i \in [1,11]$ such that:

- $[Sp_{OR}]_1(i)$ is a boolean value equal to 1 if a topological relation s_i ($i \in [1,5]$) relates the IOs in one of the n pairs selected by the user and all other boolean values are null ($[Sp_{OR}]_1(i)=0 \ \forall \ i \in [6,11]$).
- $[Sp_{OR}]_2(i)$ is a boolean value equal to 1 if a directional relation right/left s_i (i=6 or i=7) relates the IOs in one of the pairs selected by the user and all other boolean values are null.
- $[Sp_{OR}]_3(i)$ is a boolean value equal to 1 if a directional relation above/below s_i (i=8 or i=9) relates the IOs in one of the pairs selected by the user and all other boolean values are null.
- $[Sp_{OR}]_4(i)$ is a boolean value equal to 1 if a metric relation s_i (i=10 or i=11) relates the IOs in one of the pairs selected by the user and all other boolean values are null.

Generate the respective *Or* query graphs G_{spa_k}: $[IO] \rightarrow (<[Sp_{OR}]_j(i)>) \rightarrow [IO]$, $j \in [1,4]$, $i \in [1,11]$
Aggregate (join operation [Sowa 84]) CGs G_{spa_1}, G_{spa_2}, G_{spa_3} and G_{spa_4} to generate the spatial query graph G_{spa}.
Aggregate (join operation) visual semantics and spatial query graphs G_{sem} and G_{spa}. Each query (like document index representations) is indeed represented by a global CG resulting from the aggregation of CGs over the visual semantics and relational facets called **image query graph**.

5.3 Matching query and index structures

The Projection Operator. An operational model of image retrieval based on the CG formalism uses the graph projection operation for the comparison of an image query graph

and an image document graph. This operator allows to identify within a graph g_1 sub-graphs with the same structure as a given graph g_2, with nodes being possibly restricted, i.e. their types are specialization of g_2 node types. If a projection of an image query graph I_Q within an image document graph I_D exists then the image document indexed by I_D is relevant for the image query I_Q.
Formally, the projection operation $\wp : I_Q \rightarrow I_D$ exists if there is a sub-graph of I_D verifying the two following properties:

- There is a unique document concept which is a specific of a query concept, this being valid for any query concept. This property ensures that all elements describing the query are present within the image document, and their image is unique.
- For any relation linking concepts c_{Q1} and c_{Q2} of I_Q, there is the same relation between the two concepts c_{D1} and c_{D2} of I_D, such as $\wp(c_{Q1}) = c_{D1}$ and $\wp(c_{Q2}) = c_{D2}$.

However, brute-force implementations of the projection would result in exponential execution times. Based on the work in [Ounis&Pasca 98], we use an adaptation of the inverted file approach for image retrieval. We specify lookup tables associating visual semantics concepts to the set of image documents whose index contain it. Treatments that are part of the projection are performed during indexing following a specific organization of CGs which does not affect the expressiveness of the formalism. Moreover, lattices organizing spatial relations are defined by mathematical partial orders and not hard-coded, which allows fast query processing. We discuss in the next section the organization of the lattice for processing queries with OSMs.
Processing queries with OSMs. ISRs are organized within an *Or* lattice to process a query conveying a boolean disjunction of spatial relations such as "Find images with people at the left **or** at the right of buildings". This query is first translated in its graph representation (cf. section 4.2). Semantic concepts *huts* and *grass* are processed by the lattice of semantic concepts. The link between the generated OSR <P:0, T:0, D:0, C:0, C_B:0, **R:1**, **L:1**, T:0, B:0, N:0, F:0>$_{OR}$ and its equivalent ISR is not straightforward. A new category of meta-relations eliciting this link by taking into account **dominant** spatial relations (i.e. spatial relations mentioned in a query as they have a higher importance in the ordering process of ISRs within the lattice, other spatial relations are called **secondary**) shall be introduced. These concepts are index spatial meta-relations **with dominant d_{OR}**, where d_{OR} is the set of dominant spatial relations. They are supported by a vector structure s_d with eleven elements corresponding to spatial relations s_i. Values $s_d[i]_{i\in[11]}$ such that $s_i \in d_{OR}$ characterize the presence of dominant spatial relations and values $s_d[j]_{j\in[1,11]}$ such that $j \neq i$, the presence of secondary spatial relations within the spatial characterization of the considered IOs. Index spatial meta-relations **with dominant d_{OR}** are specializations of OSRs and generalizations of ISRs as far as the lattice organization is concerned. The OSR <B:0…D:1,I:0…U:1…>$_{OR}$ is related to its equivalent ISR with dominant {left, right}: <P:0, T:0, D:0, C:0, C_B:0, **R:1**, **L:1**, T:0, B:0, N:0, F:0> as highlighted in the lattice of figure 8. As a matter of fact, the most relevant images provided by the system present people at the left or at the right of buildings, i.e. people and buildings related through only dominant spatial relations. This symbolic spatial characterization is represented by the highlighted ISR (*sr*) in figure 8. Other images are composed of people either at the right or at the left of buildings with at least one additional spatial relation not mentioned in the query linking the two semantic concepts. In the lattice, ISRs representing such characterizations are descendants of *sr*. Formally, sub-lattices of index spatial meta-relations with dominant d_{OR} are partially ordered by $\leq_{OR}$:

$$\forall a,b \text{ index spatial meta-relations with dominant dOR, } a \leq_{OR} b \Leftrightarrow (a=\perp_{OR}) \vee (b=T_{OR}) \vee [(\forall i\in[1,11], s_i\in d_{OR}, (a_{[i]} = 0 \wedge b_{[i]} = 1) \vee (a_{[i]} = 1 \wedge b_{[i]} = 1)) \wedge (\forall j\in[1,11], s_j \notin d_{OR}, (b_{[i]} = 0 \wedge a_{[i]} = 1) \vee (b_{[i]} = 1 \wedge a_{[i]} = 1))]$$

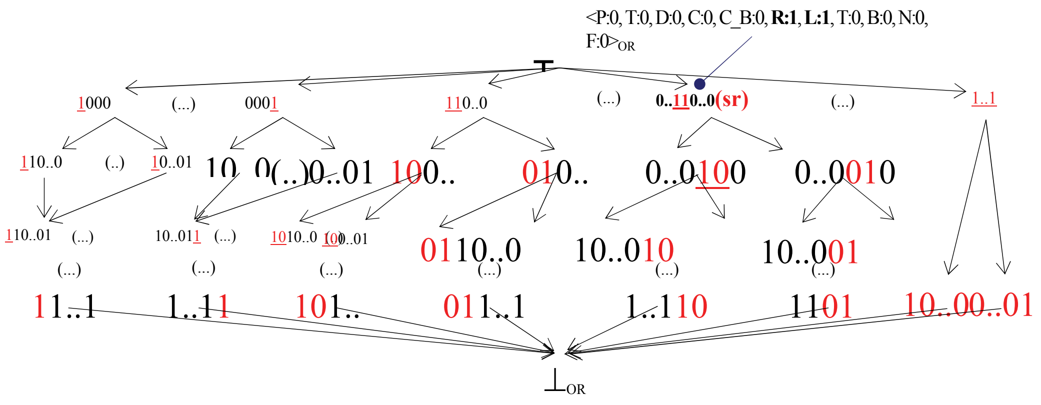


Figure 8. Lattice Processing *Or* Spatial Meta-relations

7. Conclusion

We have specified within the scope of this paper a framework combining semantics and relational (spatial) characterizations within a coupled architecture in order to address the semantic gap.
This framework is instantiated by an operational model based on a sound logic-based formalism, allowing to define a representation for image documents and a matching function to compare index and query structures.
We have specified a query framework coupling keyword-based querying with a relevance feedback module managing transparent and penetrable interactions by considering conceptual characterizations of images.
The choice of conceptual graphs as an operational model is the most natural in the sense that it holds several advantages in our application context. It indeed allows the symbolic representation of all components of a multimedia indexing and retrieval architecture: queries, index documents and matching function. Moreover its simple representation is particularly well-suited for user interaction in the framework of relevance feedback.
To stress the relevance of our approach, the theoretical contributions of this paper in the domain of image indexing and retrieval are summarized below:

- We have first proposed a neural-network based architecture for the highlighting of image objects, structures abstracting the image visual entites, and the characterization of their associated semantics.
- In the perspective of unifying the semantic and relational characterizations, we have proposed an integrated model featuring a bi-facetted organization. The visual semantics facet describes the image semantic content and is based on labeling IOs with a semantic concept. The relational facet is itself based on the relational (spatial) characterizations between pairs of image objects obtained after highlighting a correspondence process between extracted low-level information and symbolic relations.

- To overcome the limitations of the keyword-based approach to query on the image content, we have proposed a high-level relevance feedback framework, allowing in particular the relational characterization of the image objects.
- We have finally proposed a correspondence model based on the conceptual graph projection operator. Its instantiation is optimized through the use of specific data structures to boost retrieval. In particular, semantic and spatial index structures are organized in lattices defined by mathematical partial orders.

8. References

Belkhatir, M. et al. (2004). Integrating perceptual signal features within a multi-facetted conceptual model for automatic image retrieval, *ECIR*, pp. 267-282

Belkhatir, M. (2005). Combining semantics and texture characterizations for precision-oriented automatic image retrieval, *ECIR*, pp. 457-474

Belkhatir, M. et al.: A full-text framework for the image retrieval signal/semantic integration, *DEXA* (2005), pp. 113-123

Egenhofer, M. et al. (1991). Reasoning about binary topological relations, *SSD*, 143-160

Hollink, L. et al. (2004). Classification of user image descriptions. *Int. J. Hum.-Comput. Stud.* 61(5), pp. 601-626

Lu, Y. et al. (2000). A unified framework for semantics and feature based relevance feedback in image retrieval systems. *ACM MM*, pp. 31-37

Mechkour, M. (1995). EMIR^2: An Extended Model for Image Representation and Retrieval, *DEXA*, pp. 395-404

Meghini, C. et al. (2001). A model of multimedia information retrieval, *J. ACM* 48(5), pp. 909-970

Mojsilovic, A. & Rogowitz, B. (2001). Capturing image semantics with low-level descriptors, *ICIP*, pp. 18-21

Naphade, M. et al. (2006). A Large-Scale Concept Ontology for Multimedia, IEEE MultiMedia 13(3), pp. 86-91

Ounis, I. & Pasca, M. (1998): RELIEF: Combining expressiveness and rapidity into a single system. *ACM SIGIR*, pp. 266-274

Smeulders, A.W.M. et al. (2000). Content-based image retrieval at the end of the early years. *IEEE PAMI* 22(12), pp. 1349-1380

Sowa, J.F. (1984). Conceptual structures: information processing in mind and machine, Addison-Wesley publishing company

Zhou, X.S. & Huang, T.S. (2002). Unifying Keywords and Visual Contents in Image Retrieval. *IEEE Multimedia* 9(2), pp. 23-33

GNGS: An Artificial Intelligent Tool for Generating and Analyzing Gene Networks from Microarray Data

Austin H. Chen[1] and Ching-Heng Lin[2]
[1]Department of Medical Informatics, Tzu-Chi University
[1,2]Graduate Institute of Medical Informatics, Tzu-Chi University
Taiwan

1. Introduction

The completion of the Human Genome Project has been recognized as a great achievement in the study of biomedicine; the project not only provides information regarding human genes but also provides new ways to study human diseases such as cancers. High-throughput techniques, such as microarray experiments, have emerged as a method of study that measures the level of gene expression in gene networks. Since microarray experiments can produce thousands of datasets under various experimental conditions simultaneously, it is now feasible to study gene interactions and regulatory networks. How to analyze and interpret the results of these analyses, however, has become an important research area in bioinformatics.
In the study of biological cellular behavior, understanding how biological activities are governed by the relationships among genes, RNA, and proteins is a common challenge. Gene networks represent such connectivity. A gene network consists of a group of genes that interact among themselves in order to synthesize proteins. Recently, genome-wide gene expression microarray data relevant to the yeast cell cycle has been collected (Spellman et al., 1998; Cho et al., 1998; Zhu et al., 2000). Since the gene expression profile data is a record of the network interactions between the regulators and the target genes, it is possible to use this information to trace these complex relationships. A variety of computer clustering methods have been developed in order to group together genes with similar patterns of expression (Eisen et al., 1998; Tamayo et al., 1999; Tavazoie et al, 1999).
Previous efforts at modeling gene networks from high dimensional datasets have generally fallen into one of three classes, either employing Boolean networks (D'haeseleer et al., 1999; Husmeier et al., 2005), which are restricted to logical relationships between variables, or using systems of differential equations (Chen et al., 1999; Sakamoto & Iba, 2001; Thomas, 1990) to model the continuous dynamics of coupled biological reactions. The work of Friedman et al. (2000) uses Bayesian networks to analyze expression data. The statistical framework of Bayesian learning, since it deals with uncertainly, is designed for domains

with a large number of variables and for handling noisy data. Another advantage of this probabilistic approach is the ability to combine prior knowledge with the information extracted from data.

A Bayesian network is a graphical model that finds probabilistic relationships among variables (i.e. genes) of the system. Bayesian networks are popular decision support models (Cooper & Herskovits, 1992; Husmeier, 2005) because they inherently model the uncertainty in the data. In addition, Bayesian networks successfully amalgamate probability theory and graph theory to efficiently model multidimensional probability distributions by searching for independent relationships in the data (Gevaert et al., 2006; Heckerman, 1995). Other features that make Bayesian networks attractive candidates for modeling gene expression data include the ability to handle noisy or missing information, handle hidden variables, and make causal inferences. (Beal et al, 2005)

Currently, a user-friendly system that can display and analyze various gene networks from microarray experimental datasets is urgently needed. In this study, our goal is to develop a gene network generating system (GNGS) that can generate the gene networks of the yeast cell cycle from experimental microarray data as well as analyze the performance of gene networks using five different Bayesian network algorithms.

2. Methods

In this study, three kinds of datasets were used. The first two datasets are Alarm (Beinlich, 1989) and Asia (Lauritzen & Spiegelhalter, 1988) networks. These two datasets were commonly used in Bayesian networks, and the known structure of the Alarm and Asia networks are used to compare the performance of different Bayesian network algorithms.

The third dataset used in this study is S. cerevisiae cell cycle gene expression data collected by Spellman et al. (1998). This dataset contains four medium time series: 18, 24, 17 and 14 time series points for alpha, cdc15, cdc28 and elu respectively. In the assessment of a gene network, we use each of the three medium time series: alpha, cdc15, and cdc28.

After normalizing the gene expression data, we sorted these values into three classes based on Friedman's threshold value of 0.5 (Friedman et al., 2000). The data was then translated into 3 discrete values:

$$\text{Data representation} \begin{cases} \text{over-expressed} = +1 \\ \text{normal expressed} = 0 \\ \text{under-expressed} = -1 \end{cases}$$

The results were compared with a known YPL256C sub network (Dejori, 2002). In order to compare the performance of gene networks generated from different Bayesian network algorithms (Kim et al., 2004), we defined specificity and sensitivity as Formula 1 and Formula 2.

$$\text{Sensitivity} = \frac{\#\ \text{correctly estimated edges}}{\#\ \text{edges in the reference network}}$$

Formula 1: Sensitivity of gene network

$$\text{Specificity} = \frac{\#\,\text{correctly estimated edges}}{\text{all estimated edges}}$$

Formula 2: Specificity of gene network

The greater the number of correct edges, the better the sensitivity. A higher value for sensitivity and specificity indicates better performance of the gene network.

3. Bayesian network algorithms

The gene networks of the yeast cell cycle were constructed using Bayesian network algorithms from data recorded in four different microarray experimental datasets. Five computer algorithms were developed in the construction of these gene networks. Among them are the Power Constructor (PC) algorithm, Hill Climbing (HC) algorithm, Maximum-Weight Spanning Tree (MWST) algorithm, K2 algorithm and MWST+K2 algorithm. Table 1 shows a comparison of these five algorithms.

K2 algorithm	K2 is the most widely used algorithm in Bayesian network structure learning. It is well known as a general method for inferring inter-node relations in a given node group based on a complete database free of missing data [22].
MWST algorithm	The MWST algorithm was developed by Chow and Liu [4]. This algorithm searches for an optimal tree structure by using the computed mutual information as edge weights [4]. The MWST associates a weight to each connection, where each weight represents the mutual information between the two variables. When the weight matrix is created, the MWST algorithm gives an optimal tree structure.
K2+MWST algorithm	Combining the K2 and MWST algorithms provides a better quality of network by speeding the execution efficiency. A known order of nodes is first calculated using the MWST algorithm, and these results are then used in K2.
Hill climbing algorithm	The sub-optimal hill climbing method is the heuristic K2 algorithm. The method focuses solely on precision and computation time at the expense of reliability, and it mainly relies on local exploitation. The more intensive the local exploitation, the stronger the need for specialized information about the function to be minimized [15, 16].
PC algorithm	PC is one of several dependent-based algorithms. This algorithm has an intuitive basis, and under some ideal conditions, it guarantees a graph that is equivalent to a true model of the data. It can be considered a smart selection and can intelligently order the questions needed to recover a causal structure.

Table 1. Comparison of five Bayesian network algorithms

Before constructing a gene network, it is necessary to preprocess the gene expression data. The gene expression data in Spellman's experiment is first normalized into the value of log2. We then categorize these values into three classes based on Friedman's threshold value of 0.5. These classes are represented by 3 discrete values: under-expressed (-1), normal expressed (0), and over-expressed (+1). In this section, we use the K2 algorithm to demonstrate how to construct a gene network. K2 is a search and score algorithm. Initially, a node order is set. Since the quality of the network structure is sensitive to the order of the nodes, an estimation of the nodes ordering is important. At first, the initial state of every node does not include the parent nodes. We use formula 3 to appraise whether the set of parent nodes, π_i, belongs to variable i. By finding every variable (node) that maximizes $g(i,\pi_i)$, we maximize the probability of Bayesian network structure B_s belonging to data D. The algorithm will stop when there are no parent nodes could increase the score.

$$g(i,\pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!$$

Formula 3: Estimating function

As an example of how to calculate the partial conditional probability among genes, the data calculated between gene CLN2 and gene RNR3 in CDC15 conditions is computed and shown on Table 2. From Table 2, the conditional probability of P(RNR3|CLN2) can be expressed as P(-1|-1) = 0.0, P(0|-1) = 0.0, P(1|-1) = 0.556 , P(-1|0) = 0.5, P(0|0) = 0.444 , P(1|0) = 0.444 , P(-1|1) = 0.5 , P(0|1) = 0.556, P(1|1) = 0. The conditional probability is 0.0 when both RNR3 and CLN2 are under-expressed as well as 0.444 and 0.0 when both RNR3 and CLN2 are normal expressed and over-expressed. The Bayesian Gene Networks are then generated from these values using five algorithms.

		RNR3						**CDC5**		
CLN2		-1	0	1		**ACE2**		-1	0	1
	-1	0	0	0.556			-1	1	0	0
	0	0.5	0.444	0.444			0	0	0.909	0.375
	1	0.5	0.556	0			1	0	0.091	0.625
		CLB2						CLN2		
CLN1		-1	0	1		**CLB2**		-1	0	1
	-1	0	0.143	0.6			-1	0	0.25	0.625
	0	0.286	0.714	0.3			0	0.125	0.375	0.375
	1	0.714	0.143	0.1			1	0.875	0.375	0

Table 2. Partial conditional probability tables of genes in CDC15 dataset

4. GNGS system implementation

During this study, we developed a gene network generating system (GNGS) that is capable of generating the gene networks of the yeast cell cycle from experimental microarray data as well as comparing the performance of gene networks using five different Bayesian network algorithms. GNGS utilizes both MatLab's powerful processing ability and LabVIEW's dynamic interfaces in a single platform.

4.1 System architecture

GNGS is an innovative system that generates gene networks of the yeast cell cycle using Bayesian network algorithms. LabVIEW is the abbreviation for Laboratory Virtual Instrument Engineering Workbench. It is a kind of graphical programming language, or G language. The difference between the LabVIEW language and a general programming language is that LabVIEW can be written through encoded icons and can be used to construct a system. LabVIEW is an entirely graphical language which looks somewhat like an electronic schematic. It is hierarchical in that any virtual instrument that you design can be quickly converted into a module which can be a sub-unit of another virtual instrument (VI).

For example, every icon in Figure 1 has its own function. Programmers design the system by establishing connections between icons. Different data types can be expressed using color variations in the lines.

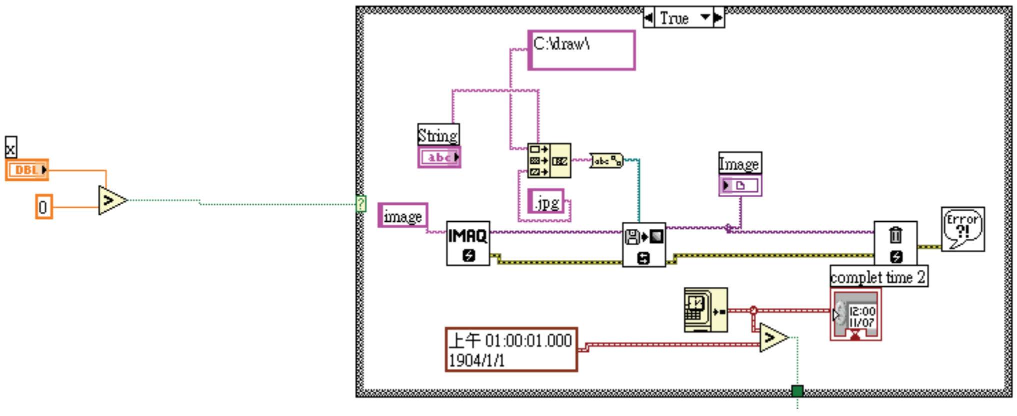


Figure 1. The VIs of displaying network function and executing time

MatLab, meanwhile, is an interactive, matrix-oriented programming language that enables us to express our mathematical ideas very concisely and directly; it considerably reduces development time and keeps code short, readable, and fully portable.

This system converts code written in MatLab and integrates the results onto the LabVIEW interface. In Figure 2 we show a Matlab scrip node (a VI provided by LabVIEW) which is the kernel that integrates the MatLab and LabVIEW languages together. The results computed from the Matlab scrip node can be seen as a VI output that transmits to another VI.

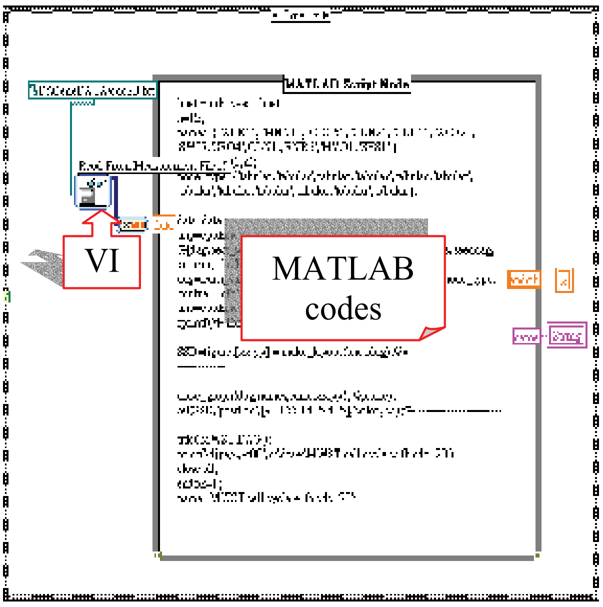


Figure 2. The Matlab scrip node generated from LabVIEW

4.2 System flow path

Figure 3 shows an overall flow path in the design of this system. A Select interface will be displayed to guide the users in running the program. By selecting the desired algorithm, GNGS will load the gene data and the system-constructed network from the selected gene dataset. After the gene network is displayed, the performance will then be calculated and shown in the summary table.

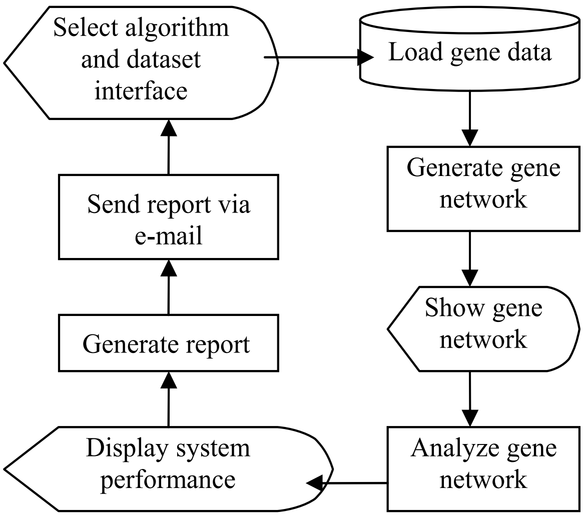


Figure 3. Design flow path of the system

5. System demonstration

The main functions of the GNGS interface include:

1. Algorithm selection section: users select the desired algorithm.
2. Network selection section: users select a desired network.
3. Dataset selection section: only displayed if the cell cycle button is clicked.
4. Execute icon: users click to run the program.
5. Clear icon: users click to clear all selections and release memory space.
6. Gene network display section: displays the resulting gene network.
7. Status slide bar section: shows the current execute status.
8. Summary table: displays summaries of users' requests.
9. Report icon: users click to generate reports that include the network graph and summary table.
10. Send icon: the system will send a report to the user through e-mail.
11. Exit icon: exits the interface.

When the users select one of the five Bayesian Network algorithms, the available network will be automatically displayed. Furthermore, if the users select cell cycle, four dataset types will then be displayed. After selecting the data set and clicking the Execute button, the light will turn to green as shown in Figure 4. The red light informs the user that the process is currently running. The slider bar on the right-hand side will show the execution status.

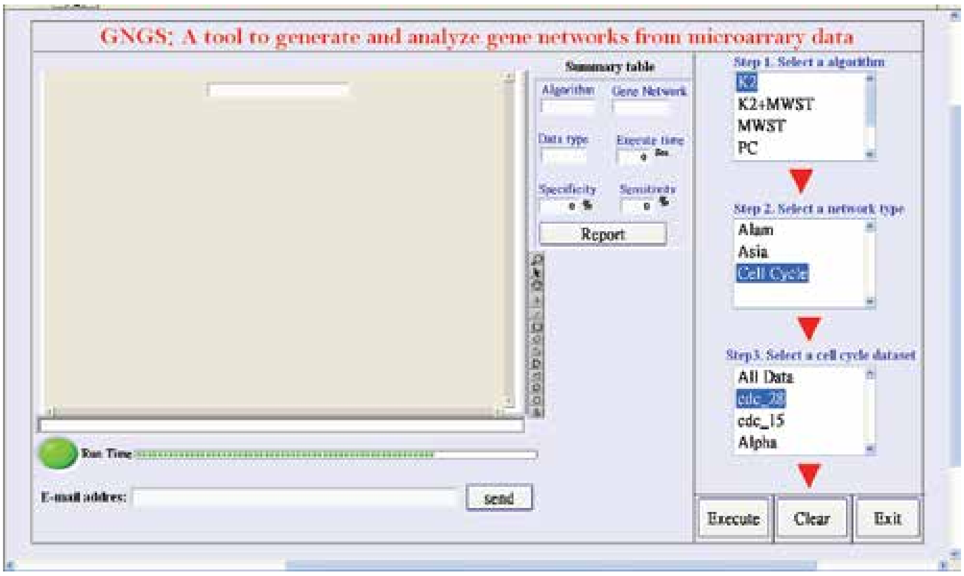
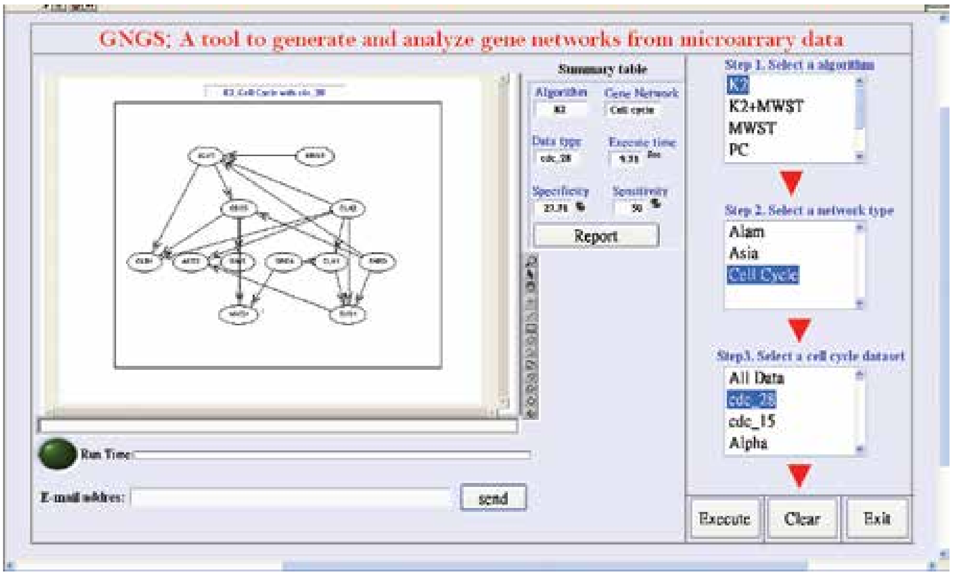


Figure 4. A screen of the system interface when the Execute icon is clicked

Figure 5 shows the final computation time and the gene network for the selected conditions. The result column will display information for the users, including algorithm, network type, dataset type, computation time, sensitivity, and specificity. The Clear button is also provided in case the users wish to clear the information. When the Clear button is clicked, the memory space will be released. Doing so assures that there are no memory overflow problems.


GNGS: A tool to generate and analyze gene networks from microarrary data

Summary table

Algorithm: K2
Gene Network: Cell cycle
Data type: cdc_28
Execute time: 9.31 Sec
Specificity: 23.31 %
Sensitivity: 50 %

Step 1. Select a algorithm: [✓ K2] K2+MWST MWST PC

Step 2. Select a network type: Alarm Asia [✓ Cell Cycle]

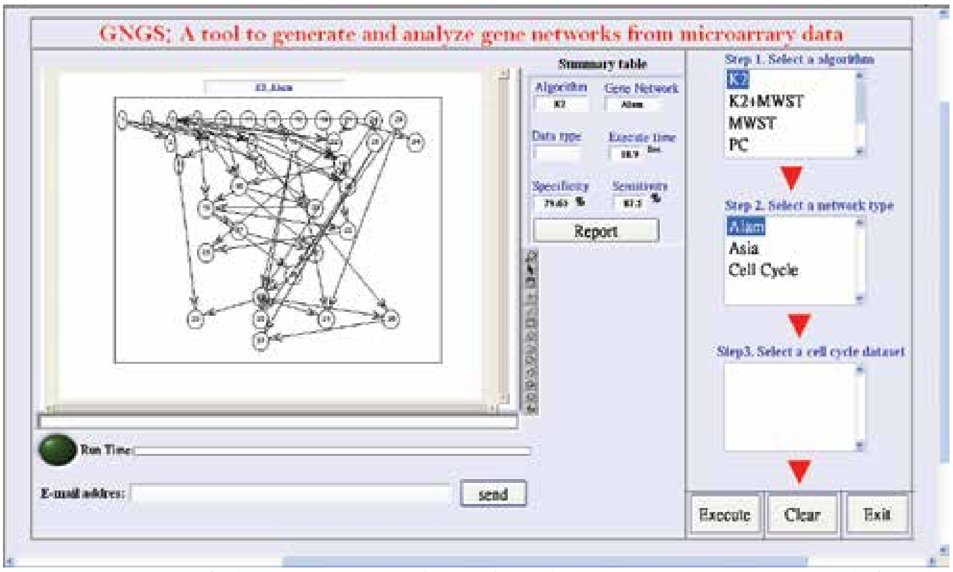
Step3. Select a cell cycle dataset: All Data [✓ cdc_28] cdc_15 Alpha

Figure 5. A screen of the system interface when the execution is finished.

6. Performance comparison

6.1 Alarm and Asia networks

Both the Alarm and Asia networks were used in this study to compare the performance for five different Bayesian network algorithms. Using the K2 algorithm as an example, the results of the Alarm and Asia network comparisons are shown in Figures 6 and Figure 7, respectively. In Figure 7, two structures are displayed: a known Asia network structure on the left and a K2-generated network on the right.


GNGS: A tool to generate and analyze gene networks from microarrary data

Summary table

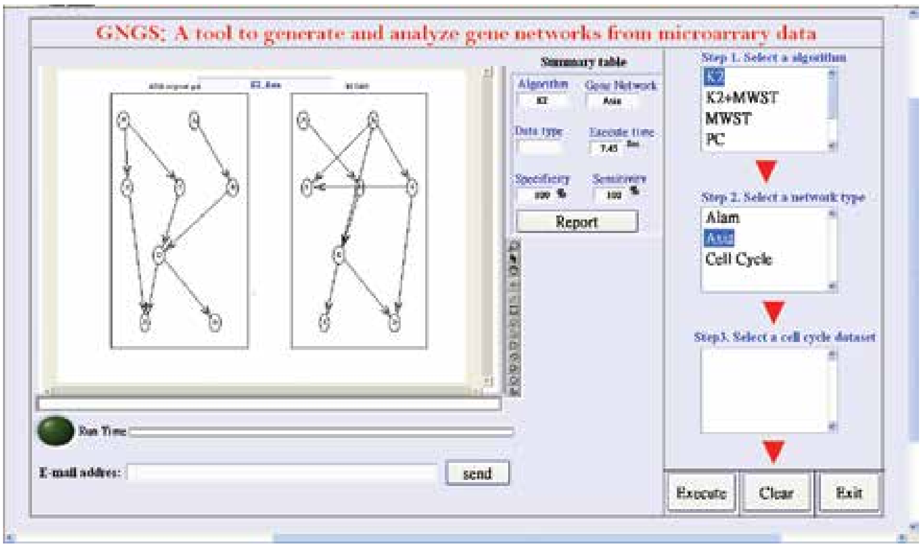
Algorithm: K2
Gene Network: Alarm
Data type:
Execute time: 18.9 Sec
Specificity: 79.43 %
Sensitivity: 83.3 %

Step 1. Select a algorithm: [✓ K2] K2+MWST MWST PC

Step 2. Select a network type: [✓ Alarm] Asia Cell Cycle

Step3. Select a cell cycle dataset:

Figure 6. A screen of the system interface after the execution is completed for the K2 algorithm and the Alarm network.


GNGS: A tool to generate and analyze gene networks from microarray data

Summary table

Algorithm: K2
Gene Network: Asia
Data type:
Execute time: 7.45 Sec
Specificity: 100 %
Sensitivity: 100 %

Step 1. Select a algorithm: [✓ K2] K2+MWST MWST PC

Step 2. Select a network type: Alam [✓ Asia] Cell Cycle

Step3. Select a cell cycle dataset

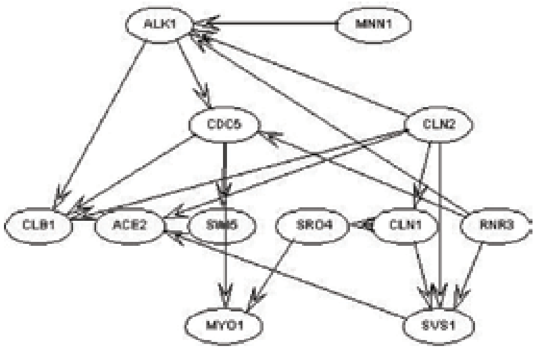
Figure 7. A screen of the system interface after the execution is completed for the K2 algorithm and the Asia network.

6.2 Gene networks of the yeast cell cycle

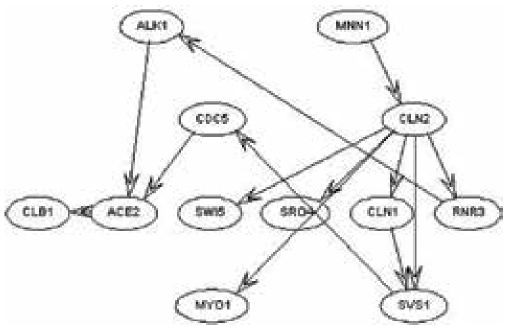
GNGS can generate gene networks based on the selected algorithm and dataset. In Figure 8, we show two gene networks generated from two experimental microarray datasets: one from the cdc_28 dataset and one from the cdc_15 dataset.

6.3 Comparison of computer execution time

In this section we compare the computer execution time for six datasets based on the K2 algorithm. The Alarm network, as expected, had the longest execution time at 28.1 seconds because it had the largest amount of data; gene networks from four cell cycle datasets all had an execution time of less than 1 second (Figure 9). The quantity of data within the dataset can affect the computer's execution time. Thus, we compared the system's execution time for five algorithms based on the same dataset. The times for K2, MWST, and K2+MWST were all less than one second. More complex search algorithms, such as the PC and HC algorithms, had a longer computer execution time; these, however, were still less than one minute (Figure 10).



(a)



(b)

Figure 8. Gene networks generated from the K2 algorithm by two different microarrary datasets: (a) the cdc_28 dataset, and (b) the cdc_15 dataset.

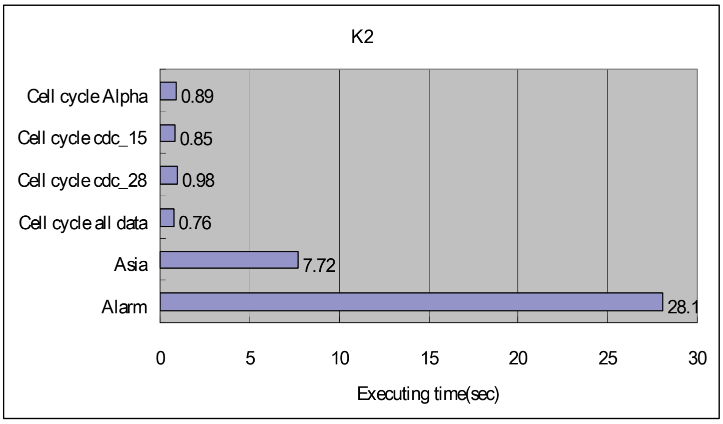


Figure 9. Comparison of execution time for 6 datasets based on the K2 algorithm

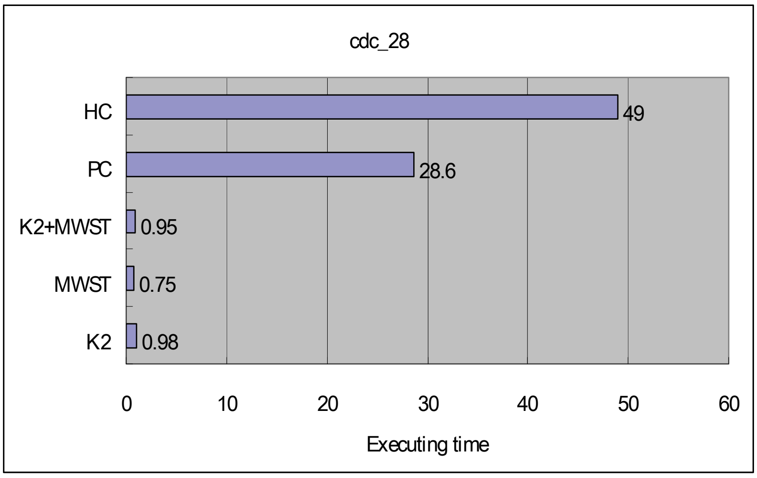


Figure 10. Comparison of execution time for 5 algorithms using cdc_28 dataset.

6.4 Comparison of sensitivity and specificity

By comparing the sensitivity and specificity of gene networks generated from six datasets using the K2 algorithm, it was found that the gene network constructed from the Asia dataset had the best performance (Figure 11). Both sensitivity and specificity were at 100 percent accuracy. The gene networks constructed from the cell cycle datasets, however, produced approximately 50 percent sensitivity.

In addition, we compared gene networks constructed based on five algorithms from the same dataset. It was notable that the HC algorithm required the longest computer time (Figure 10 and 12).

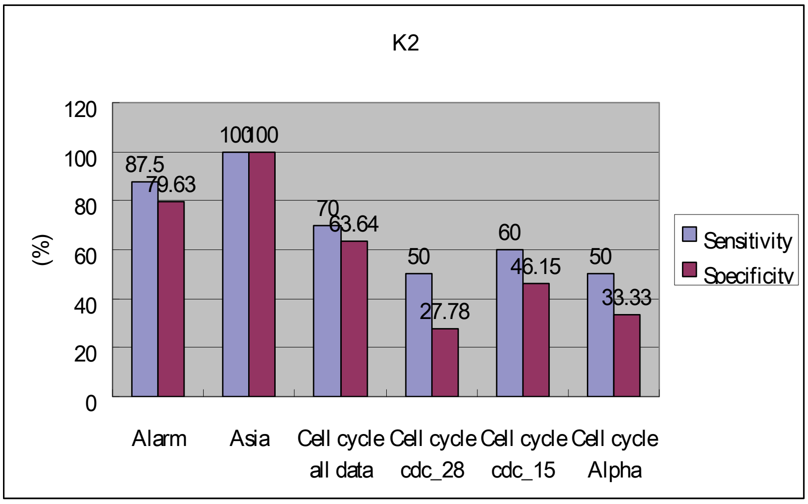


Figure 11. Comparison of sensitivity and specificity for 6 datasets based on the K2 algorithm

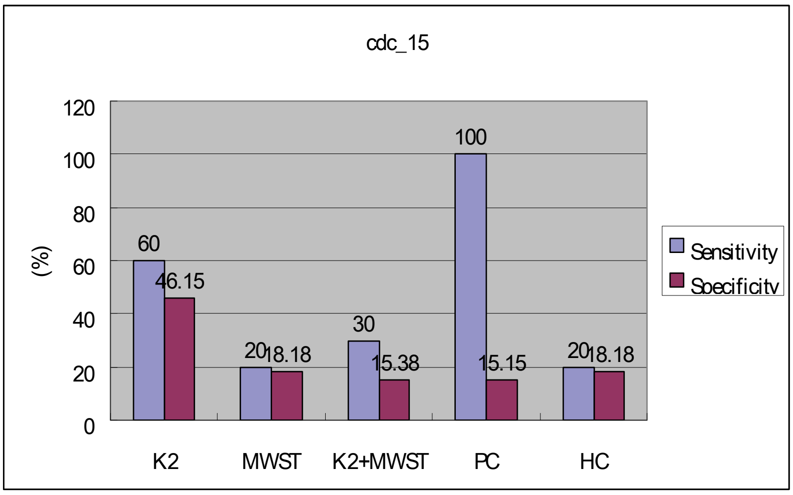


Figure 12. Comparison of sensitivity and specificity for 6 algorithms based on cdc_15 data set

6.5 Summaries of system performance

Finally, the system performance for all gene networks generated from the four experimental microarray datasets using five Bayesian network algorithms is summarized in Table 3. It was noted that the computer execution times for K2, MWST, and K2+MWST were all less than one second. Even for the more complicated operations such as the PC and HC algorithms, the execution time was still less than one minute.

A comparison of characteristics for each algorithm and its performance is summarized in Tables 3. Among these six algorithms, K2 has the best performance in terms of execution time, sensitivity, and specificity. In essence, the more nodes within a network structure (such as the Alarm network), the more run time is needed. This study found that the search and score method (K2) is the best strategy to find the optimal gene network due to its excellent performance and short execution time. The GNGS system is capable of running these algorithms, displaying the resulting networks, and analyzing system performance simultaneously.

Algorithm	Network	Data type	Executing time(sec)	Sensitivity	Specificity
K2	Alarm	-	28.1	87.5	79.63
	Asia	-	7.72	100	100
	Cell cycle	All data	0.76	70	63.64
		cdc_28	0.98	50	27.78
		cdc_15	0.85	60	46.15
		Alpha	0.89	50	33.33
MWST	Alarm	-	4.48	32.61	41.67
	Asia	-	7.92	62.5	71.43
	Cell cycle	All data	0.73	30	27.27
		cdc_28	0.75	20	18.18
		cdc_15	0.75	20	18.18
		Alpha	0.75	18.18	18.18
K2+MWST	Alarm	-	26.2	36.96	26.98
	Cell cycle	All data	0.9	30	27.27
		cdc_28	0.95	30	23.08
		cdc_15	0.9	30	15.38
		Alpha	1	30	15.38
PC	Asia	-	20.6	75	100
	Cell cycle	All data	20.7	100	15.15
		cdc_28	28.6	100	15.15
		cdc_15	28.9	100	15.15
		Alpha	28.5	100	15.15
HC	Asia	-	57	37.5	30
	Cell cycle	All data	55.1	20	18.18
		cdc_28	49	30	33.33
		cdc_15	51.4	20	18.18
		Alpha	52.7	30	27.27

Table 3. Summaries of system performance for all gene networks generated from four experimental microarray datasets using five Bayesian network algorithms.

7. Conclusion

In this paper, we have described a novel method to approach the study of gene networks. Firstly, we have developed and written five Bayesian network algorithms to construct gene networks of the yeast cell cycle based on four different microarray datasets. Secondly, we have implemented a gene network generating system that is more user-friendly. GNGS is capable of generating gene networks of the yeast cell cycle from experimental microarray data and comparing the performance of gene networks using five different Bayesian network algorithms. Our system utilizes both the powerful processing abilities of MatLab and the dynamic interface of LabVIEW in a single platform. Thirdly, we have compared the performance of each algorithm through measures such as execution time, sensitivity, and specificity for all five algorithms based on four different datasets.
In the near future, we intend to further improve performance by utilizing dynamic Bayesian network algorithms that more accurately reflect living cells' dynamic behavior. Our approach will then be used to explore the gene networks of human cells based on the microarray datasets of human cancers.

8. References

Beal et al (2005) A Bayesian approach to reconstructing genetic regulatory networks with hidden factors, *Bioinformatics, Vol 21, No. 3, 349 – 356*

Beinlich, I.A., Suermondt, H.J., Chavez, R.M., and Cooper, G.F. (1989) The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks, *Technical Report KSL-88-84*, Knowledge Systems Lab, Medical Computer Science, Stanford University.

Chen, T., He, H.L., Church, G.M. (1999) Modeling Gene Expression with Differential Equations, *Proc. of Pacific Symposium on Biocomputing*, pp. 29-40.

Cho,R.J. et al. (1998) A genome-wide transcriptional analysis of the mitotic cell cycle". Mol. *Cell*, 2, 65–73.

Chow, C., and Liu, C., (1968) Approximating discrete probability distributions with dependence trees, *IEEE Transactions on Information Theory*, 14(3), 462–467.

Cooper, G.F., Herskovits, E. (1992) A Bayesian method for the induction of probabilistic networks from data. *Mach. Learning J.* 9, 309–347.

Dejori, J. (2002) Analyzing Gene-Expression Data with Bayesian Networks, MS Thesis, *Elektro- und Biomedizinische Technik Technische Universit¨*at Graz. 2002.

D'haeseleer, P, Liang, S, and Somogyi, R. (1999) Tutorial: Gene Expression Data Analysis and Modeling, *Pacific Symposium on Biocomputing '99* (PSB'99).

Eisen,M.B., Spellman,P.T., Brown,P.O., and Bostein,D. (1998) Cluster analysis and display of genome-wide expression patterns., *Proc. Natl Acad. Sc*i., USA, 95, 14863–14868.

Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000) Using Bayesian networks to Analyze Expression data, *Journal of Computational Biology*, 7, 601-620.

Gevaert,O. et al. (2006) Predicting the prognosis of breast cancer by integrating clinical and microarray data with Bayesian networks. *Bioinformatics*, 22, 184–190.

Heckerman, D., Geiger, D. and Chickering, D.(1995) Learning Bayesian networks: The combination of knowledge and statistical data, *Machine Learning*, 20(3), 197-243.

Husmeier,D., Dybowski,R. and Roberts,S., eds (2005) Probabilistic modelling in bioinformatics and medical informatics. *Springer-Verlag,* London, UK.

Kim, S., Imoto, S., and Miyano, S. (2004) Dynamic Bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression data, *Biosystems*, 75, 57-65.

Lauritzen, S.L., and Spiegelhalter, D.J. (1988) Local computations with probabilties on graphical structures and their application to expert systems, *J. Royal Statistical society B*, 50:154-227.

Ovalle-Mart´ınez, F.J., Gonz´alez, J.S., and Stojmenovi´c, I., (2004) A parallel hill climbing algorithm for pushing dependent data in clients-providers-servers systems, *Mobile Network and Applications*, 9:257–264.

Renders, J.-M. and H. Bersini (1994) Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways, Proceedings of the First IEEE International Conference on Evolutionary Computation, pp. 312-317. IEEE Press.

Sakamoto, E. and Iba, H. (2001) Inferring a System of Differential Equations for a Gene Regulatory Network by using Genetic Programming, IEEE Press, *Congress on Evolutionary Computation*, pp.720-726,.

Spellman, P.T., et al. (1998) Comprehensive Identification of cell cycleregulated genes of the yeast saccharomyces cerevisiae by microarray hybridization, *Molecular Biology of the Cell*, 9, 3273-3297.

Tamayo, P. et al. (1999) Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation, *Proc. Natl Acad. Sci.*, USA, 96, 2907–2912.

Tavazoie, S. et al. .(1999) Systematic determination of genetic network architecture. *Nat. Genet*., 22, 281–285.

Thomas H. Cormen, Charles E. Leiserson, and Ronald R. Rivest (1990) Introduction to Algorithms., *MIT* Press.

Wright, R. and Yang, Z. (2004) Privacy-preserving Bayesian network structure computation on distributed heterogeneous data, In 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), Seattle, WA, USA.

Zhu, G. et al. (2000) Two yeast forkhead genes regulate the cell cycle and pseudohyphal growth." *Nature*,406, 90–94.

Preferences over Objects, Sets and Sequences

Sandra de Amo and Arnaud Giacometti
Universidade Federal de Uberlândia
Université de Tours
Brazil
France

1. Introduction

Recently, a lot of interest arose in the artificial intelligence and database communities concerning the topic of preference elicitation, modelling and reasoning. In fact, due to the huge amount of information users are faced up to daily, the development of formalisms allowing preference specification and reasoning turns out to be an essential task. A lot of work has been done in this area so far (Boulilier et al., 2004); (Brafman et al., 2006a); (Chomicki, 2003); (Kießling, 2002); (Wilson, 2004). Most of this work focus on specifying and reasoning with preferences over objects in some universe *U*. In most applications, mainly those related to the database field, one deals with huge set of objects, which makes unfeasible for the users to specify their preferences in a *quantitative* way, that is, by explicitly associating to each object (or tuple) *o* a number *pref(o)* standing for her *degree of preference* concerning this object. A *qualitative* framework for expressing preferences over objects is more suitable in this case. The user is asked to provide a set of statements or rules which express her generic preferences over the attribute values of the objects. For instance, the user can express her preferences about films by stating that (1) concerning comedies, she prefers those from Woody Allen to those from Nanni Moretti (2) concerning Nanni Moretti's movies, she prefers comedies to dramas. Such frameworks, besides providing a compact way for expressing preferences, are also supposed to derive an explicit preference ordering over the objects, given the compact specification provided by the user, and produce an algorithm to determine the most preferred objects according to this ordering.
Some recent research on preference elicitation and reasoning has focused on preference over more complex entities, like *sets* of objects (Brafman et al., 2006b); (des Jardin & Wagstaff, 2005). Indeed, in many situations, instead of selecting a most preferred object, one may be interested in selecting a best *set* of objects whose components satisfy certain criteria of diversity and mutual compatibility. For instance, in the creation of a film festival program, a criteria for a "good" program could be *"a program including a comedy is better than a program which doesn't include one"*.
However, more complex entities other than simple sets of objects have been appearing in recent applications. For instance, in the design of a web page, the developer can take into account user preferences about hyperlink structures (trees). In our example of the film festival program, an optimal program should not only be characterized by the quality, diversity and compatibility of its components but also by the *ordering* in which each film is

presented in the program. So, it is natural to think about preference elicitation and reasoning over *structures* rather than merely over simple objects or non-structured sets of objects.
In this chapter, we cover with some details some classical and important formalisms to specify preferences over objects and sets of objects and we address the problem of specifying and reasoning with preferences over *sequences* of objects. The material we present here addressing this topic is an extension of our previous work (de Amo & Giacometti, 2007). Preferences over sequences of objects naturally appear when a decision maker is faced to the problem of producing an optimal sequence of objects. The following example illustrates the kind of preference statements we will deal with.
Example 1 Let us suppose a decision maker who works on the creation of a program for a film festival. Based on his past experiences on film festivals, there are some rules he thinks are crucial to the success of such an event.

1. For comedies, it is better to choose those by Woody Allen than those by Nanni Moretti. Concerning Nanni Moretti's movies, comedies are better than dramas.
2. It is better to start the festival by presenting a comedy.
3. If the previous film was a comedy, then it is better to follow it by a drama. However, if the previous film was a drama, then it is better to follow it by a comedy, unless it is a film by Nanni Moretti, in which case, it is better to follow it by another drama.
4. If there is a drama in the program, then it is better to present a comedy sometime before it.

We introduce the logic framework **TPref** allowing preference elicitation and reasoning over sequences of objects as well as an algorithm to yield the most preferred sequences satisfying a given set of *temporal* constraints. Our elicitation procedure consists in obtaining from the user (1) a set of *temporal conditions* which affects her preferences over sequences of objects and (2) a set of statements or rules involving these temporal conditions, which express her preferences. The four statements illustrated in Example 1 are preference statements we treat in this paper.
After preference elicitation, the statements provided by the user are translated into formulae of the logic **TPref**. Our formalism, which is based on Propositional Temporal Logic (PTL), generalizes the language introduced in (Wilson, 2004) for expressing preference over single objects. We show a procedure to decide the consistency of a set of statements in the *past* fragment of the logic **TPref**, that is, if a set of statements Φ (a compact preference representation) derives an explicit preference ordering $>_{\phi}$ over sequences of objects. We discuss the difficulties for using this same idea in proving consistency in the general case of preference statements involving past and future conditions. Finally, we provide an algorithm for producing the best sequences of objects given a set of temporal preference statements Φ.

1.1 Related work

The research literature on preference reasoning and elicitation over objects is extensive. The approach of CP-Nets (Boutilier et al., 2004) uses a very simple graphical model which captures users qualitative conditional preference over objects, under a *ceteris paribus* semantics. The order on objects induced by a CP-Net is rather restrictive, due mainly to the ceteris paribus semantics and also by the fact that all attributes are equally important where comparing two objects. The approach of TCP-Nets (Brafman et al., 2006a) generalizes the

CP-Nets by introducing the ability of expressing a relative importance and conditional relative importance of object attributes. Thus, a TCP-Net is a more refined tool for comparing objects than CP-Nets. The approach introduced in (Wilson, 2004) uses a logical framework for expressing conditional preference statements. It consists of a formalism in the same lines of CP-Nets but with a richer language allowing to express not only the usual CP-Nets ceteris paribus statements but also TCP-Nets statements and more general conditional statements (called *stronger conditional statements*). The temporal conditional preference statements we introduce in Section 4 for specifying preferences over sequences of objects is a generalization, in the temporal context, of the stronger conditional statements of (Wilson, 2004). The conditions in a stronger conditional statement can be viewed as a propositional logic formula. In our approach for specifying preferences over sequences of objects, the conditions are propositional *temporal* logic formulae.

In the database area, the problem of enhancing well-known query languages with preference features has been tackled in several recent and important works in the area. In (Chomicki, 2003), a simple logical framework is proposed for expressing preferences. Preferences are expressed by *preference formulae*. These formulae are incorporated into relational algebra and into SQL, through the operator *winnow* parameterized by a preference formula. (Kießling, 2002) introduced Preference SQL which extends SQL by a preference model based on strict partial orders. Several built-in base preference constructors are proposed. The optimizer uses an efficient rewriting procedure which transforms preference queries into standard SQL queries.

Recent work on preference modelling in AI has focused on *sets* of objects instead of single objects. In (Brafman et al., 2006b), a language for specifying *qualitative* preferences over sets is introduced. The language allows users to express preferences over sets of objects taking into account a class of basic properties which affect their choice. It is shown that a set-preference statement specified in this language can be transformed into a conditional preference statement over attributed objects. The language introduced in (des Jardins&Wagstaff, 2005) allows *quantitative* preference specification over sets of objects. It supports two important preference notions: *diversity* and *depth*. *Diversity* specifies the amount of variability among objects in a set, and *depth* specifies preferred feature values.

Most work on *temporal reasoning with preferences* is related to automated plannning. Preferences concerns the relative execution times of a set of events $\{e_1, \ldots, e_n\}$ (Khatib, 2001); (Kumar, 2007). A preference statement in such an *explicit* temporal framework may establish for instance that event e_i must be scheduled between x_i and y_i seconds before event e_j.

Propositional Temporal Logic (PTL) was introduced in (Prior, 1997) as a formal system for specifying and reasoning with paralell programs. Recently, PTL has been used in the automated planning context, as a formalism to specify "good" executing plans (which can be viewed as sequences of state transitions). In (Bacchus et al., 1996), PTL has been used in the automated planning context where actions depend on past and current states. The problem considered there is of rewarding temporally extended behaviors, that is, rewarding sequence of actions (or state transitions) achieving a predefined goal. Rewards are associated to properties that sequences must satisfy. Such properties are expressed by PTL formulae. In (Bienvenue et al., 2006), a formalism based on temporal logic and situation calculus was introduced in order to express *qualitative preferences* about executing plans. Such formalism allows to specify, to reason and to generated preferred plans. This approach generalizes the one proposed in (Son & Pontelli, 2006), which also uses temporal logic for expressing preferences over executing plans with an implementation using answer-set programming. At the best of our knowledge, there are no work treating qualitative conditional preferences elicitation and reasoning over *sequence of attributed objects* in the lines of the CP-Net formalism. The approach we propose in this chapter is a first step towards incorporating a formalism for *reasonning* with preferences over sequences of objects into a temporal relational query language, and so, building a bridge between the two disciplines (AI and Temporal Databases), in the lines of which has been done in (Endres & Kießling, 2006), where a method for transforming TCP-Nets queries into database preference queries has been proposed.

Chapter Organization. This chapter is organized as follows. In Section 2, we present three classical approaches for preference specification over *objects*, the CP-Nets, the TCP-Nets and the strong conditional statements of (Wilson, 2004). We discuss important problems related to this topic, such as finding the most preferred objects, comparing objects (dominance queries) and ordering objects (ordering queries). We describe the third approach (Wilson, 2004) with more details since it constitutes a necessary background for our work on sequences of objects. In Section 3, we present a simple approach for specifying preferences over *sets of objects*. In Section 4, we present our approach for eliciting and reasoning with preference over *sequences of objects*. In this Section, we introduce the syntax and semantics of the language **TPref** allowing to express preferences over sequence of objects, we show how to test the consistency of a set of statements Φ in **TPref**, and discuss its complexity. Besides, we present an algorithm to produce the optimal sequences satisfying a set of simple temporal constraints.

2. Preferences over objects

In most AI applications involving the ability of making decisions, users are required to compare different alternatives and must be able to choose those which better conform to their needs or personal preferences. Thus, such applications must support the ability of automate the preference elicitation process. In this section we will present three important approaches for representing and reasoning with preferences over objects. The first approach is based on a graphical model focusing on the notion of conditional preferential independence. The second approach is also based on a graphical model and generalizes the first one. The third approach is quite general and is based on a logical framework allowing users to express their preferences through a set of rules.

2.1 Conditional preference networks (CP-Nets)

The graphical model we describe in this section was introduced in (Boutilier et al., 2004) and is similar to a Baysesian Network (Pearl, 1988) from a syntactical point of view. Nevertheless both models differ with respect to their semantics. The model we present here, called *Conditional Preference Networks (CP-Nets)* uses a graph in order to capture statements of *qualitative* conditional preference independence. The semantics of the model is based on the *ceteris paribus* semantics which has been largely exploited in the AI field in the past (Doyle et al., 1991). Other approaches for representing and reasoning with preferences have employed graphical representations of preference relations such as (Bacchus & Grove, 1995) and (Bacchus and Grove, 1996), but with a different semantics.

In the CP-Net preference model, the user is required to specify, for any specific attribut A of interest, which other attributes can influence her preferences for values of A. For each instantiation of the relevant attributes for A (the parents of A in the graphical representation) the user must specify her preference ordering over values of A according to the values of its parents. For instance, let us consider a set of objects with attributes A,B,C,D and let us suppose that preference over attribute A depends on attributes B and C. So, the user may specify that *if the value of B is b_1 and the value of C is c_2, and everything else is equal then she prefers a_2 to a_1 as a value for attribute A*. Based on this preference rule, the user can decide that between two objects $o_1 = (a_1, b_1, c_2, d_1)$ and $o_2 = (a_2, b_1, c_2, d_1)$ she prefers object o_2 to object o_1. On the other hand, this rule cannot allow her to decide that object o_2 is preferred to object $o_3 = (a_1, b_1, c_2, d_2)$, since the values of the attribute D in both objects are different. The *ceteris paribus semantics* (*everything else being equal*) imposes that we can only compare objects according to a given preference rule r if the objects have the same values on the attributes not appearing in r.

Notation. We suppose a set $V = \{X_1, X_2, ..., X_n\}$ of attributes. For each attribute $X \in V$, we denote by $\mathbf{dom}(X)$ the finite set of values of X (the domain of X). For $Z = \{Z_1, ..., Z_m\} \subseteq V$ we denote by $\mathbf{dom}(Z)$ the set $\mathbf{dom}(Z_1) \times \mathbf{dom}(Z_2) \times ... \times \mathbf{dom}(Z_m)$. If $Z = V$, we denote by $\mathcal{O}$ the set $\mathbf{dom}(Z)$. The elements of $\mathcal{O}$ are called *objects, tuples or outcomes*. If $o = (x_1, ..., x_n)$ is an object, we denote by $o[X_i]$ the element $x_i \in \mathbf{dom}(X_i)$. If $Z = \{Z_1, ..., Z_m\} \subseteq V$, we denote by $o[Z]$ the tuple of elements $(o[Z_1], ..., o[Z_m])$. Sometimes we abbreviate this tuple by $\mathbf{z}$.

Definition 1 (CP-Net Preference Model) A *CP-Net* over a set of attributes V is a directed graph $\mathcal{N} = (V , E)$ where each node $X \in V$ is annotated with a conditional preference table (CP table) CPT(X). Each CP table CPT(X) associates a total order $\succeq_{\mathbf{u}}$ with each instantiation $\mathbf{u}$ of the attributes which are parents of X in the graph.

The following example illustrates the concept of CP-Net as a formalism for specifying user's preferences.

Example 2 Let V = {Director (D), Genre (G) }, $\mathbf{dom}(D)$ = {Woody Allen (w), Nanni Moretti (n), Hitchcock (h)}, $\mathbf{dom}(G)$ = {comedy (c), drama (d), thriller (t)}. Let us suppose that I strictly prefer comedies to dramas and thrillers to comedies but my preference about film directors is conditioned to the film genre: I prefer Nanni Moretti's dramas toWoody Allen's dramas, and Woody Allen's dramas to Hitchcock's dramas. However, I prefer Woody Allen's comedies to Nanni Moretti's comedies and Nanni Moretti's comedies to Hitchcock's comedies. On the other hand, for thrillers I largely prefer Hitchcock's ones than Woody Allen's. But if I had to choose between a Woody Allen's thriller and a Nanni Moretti's thriller I would choose a Woody Allen's thriller. These preference rules can be expressed by the CP-Net depicted in Figure 1(a).

A CP-Net aims at capturing a *preference ordering* (a total order) over the objects in $\mathcal{O}$. Thus, the semantics of a CP-Net is defined as the set of preference orderings which are consistent with the preference constraints imposed by the given CP-Net. In Figure 1(b) one represents by thin arrows the relationships between objects which are entailed by the CP-Net of Figure 1(a). An arrow from object o to object o' means that $o \succeq o'$. The arrows resulting from transitivity (e.g. from (t, h) to (d, n)) are not showed in the figure. The thick arrows are not entailed by the CP-Net of Figure 1(a) but are consistent with it (see the discussion following Theorem 1 below).

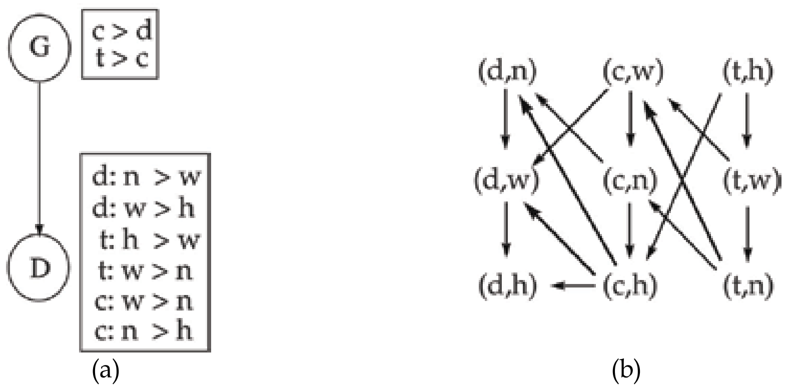


Fig. 1. (a) A CP-Net $\mathcal{N}$ (b) A preference ordering satisfying $\mathcal{N}$

Definition 2 (Satisfiability of a CP-Net) Let $\mathcal{N}$ be a CP-Net over the set of attributes V, $X \in V$ and $U \subset V$ the set of parents of X in $\mathcal{N}$. Let Y be the set of attributes other than X and its parents. Let $\succeq_{\mathbf{u}}$ be the ordering over **dom**(X) imposed by the CPT(X) for a given instantiation **u** of the attributes in U. We say that a preference ordering $\succeq$ over $\mathcal{O}$ *is compatible with* $\succeq_{\mathbf{u}}$ iff for all instantiations **y** of the attributes in Y we have $\mathbf{y}x\mathbf{u} \succeq \mathbf{y}x'\mathbf{u}$ iff $x \succeq_{\mathbf{u}} x'$. A preference ordering $\succeq$ *satisfies* the CP table CPT(X) iff it is compatible with $\succeq_{\mathbf{u}}$ for any instantiation **u** of the attributes in U. We say that the preference ordering $\succeq$ *satisfies* the CP-Net $\mathcal{N}$ iff it satisfies all the CP tables of $\mathcal{N}$. A CP-Net $\mathcal{N}$ is *satisfiable* iff there exists some preference ordering $\succeq$ satifying it.

In Figure 1(b) it is depicted a preference ordering satisfying the CP-Net of Figure 1(a). The following theorem guarantees that for acyclic CP-Nets it is possible to build an ordering satisfying it.

Theorem 1 Every acyclic CP-Net is satisfiable.

The detailed proof of Theorem 1 can be found in (Boutilier et al., 2004). The ordering given by this theorem is built by induction on the number of attributes in the CP-Net and uses the topological ordering on these attributes induced by the acyclic graph. The preference ordering depicted in Figure 1(b) is obtained by using the construction of Theorem 1. The thick arrows are specific to this particular ordering. They are built respecting the ordering given in the CP table CPT(G). A preference ordering satisfying an acyclic CP-Net is not unique in general. For instance, if we consider an arrow going from (d, n) to (c, h) in Figure 1(a) instead of the opposite arrow depicted in this figure, and we keep the other arrows, we obtain another ordering satisfying $\mathcal{N}$.

Best Outcomes. Given an acyclic CP-Net $\mathcal{N}$, the task of determining the best outcomes with respect to the preference orderings satisfying $\mathcal{N}$ is very simple. Even if the preference ordering satisfying $\mathcal{N}$ is not unique, surprisingly, the best outcome is unique and independs on the particular preference ordering satisfying $\mathcal{N}$. The algorithm for building this unique best outcome consists in sweeping through the graph from ancestors to descendents instantiating each attribute to its most preferred value given the instantiation of its parents. We describe the process of determining the best outcome in the following example.

Example 3 (Producing the best outcome determined by a CP-Net) Let us consider the CP-Net of Example 2. We begin by choosing the best value for attribute G (the attribute with no ancestors). This best value is t. Next, we take the children of attribute G. In our case, we have only one child, the attribute D. For G instantiated as t, the best value for the attribute D is h. Then, the best outcome is the object (t, h), that is, the most preferred movie is a Hitchcock's thriller.

This process of sweeping through the graph from ancestors to descendents and instantiating the attributes with the most preferred values given the instantiation of their parents is called *forward sweep*. The following theorem guarantees that this procedure produces the best outcome. The proof can be found in (Boutilier et al., 2004).

Theorem 2 Let $\mathcal{N}$ be an acyclic CP-Net. The best outcome with respect to any preference ordering satisfying $\mathcal{N}$ is unique and is produced by the forward sweep procedure.

Discussion. The CP-Net model for preference reasoning is not restricted to acyclic graphs. The advantage of considering acyclic CP-Nets is that the acyclicity of the graph implies that the model is consistent, that is, the CP-Net induces a preference ordering over the objects. If the graph is cyclic, the existence of such preference ordering is not guaranteed. In (Domshlak & Brafman, 2002) some initial results on consistency testing for cyclic CP-Nets were presented. More recently (Prestwich et al., 2005) showed that the optimal outcomes of an unconstrained (and possibly cyclic) CP-Net are the solutions of a set of hard constraints. They proposed a new algorithm for finding optimal outcomes which makes use of hard constraint solving. This new algorithm works even for cyclic CP-Nets. Besides, it works also with any preference formalism which produces a preorder over the outcomes. Another aspect which has to be considered is the constraint enforcing that in each CP table CPT(X), the domain **dom**(X) is totally ordered. The general definition of a CP-Net allows an arbitrary total *preorder* over **dom**(X), that is, the antisymmetric property is not required to be satisfied ($a \succeq b$ and $b \succeq a$ do not imply $a = b$). The difficulty with such general CP-Nets is that consistency is not verified in general. In (Boutilier et al., 2004) it is proved that consistency can be guaranteed if some special conditions are verified by the acyclic CP-Net.

Besides the problem of finding the best outcomes determined by a CP-Net $\mathcal{N}$, two other problems are particularly important: the dominance problem and the ordering problem. Both problems involve the task of comparing two objects o and o'. The first problem asks if $\mathcal{N}$ can deduce $o \succeq o'$(denoted by $\mathcal{N} \;|= o \succeq o'$). That is, it asks if *for all preference orderings* $\succeq$ *consistent with* $\mathcal{N}$ *it is true that* $o \succeq o'$. The second problem asks if the CP-Net is incapable of deducing $o' \succeq o$ (denoted by $\mathcal{N} \;|\neq o' \succeq o$). That is, it asks if *there exists a preference ordering* $\succeq$ *consistent with* $\mathcal{N}$ *such that* $o \succeq o'$. The second problem is easier than the first one. It can be proven that for acyclic CP-Nets, the complexity of determining the truth of at least one of the orderings queries $\mathcal{N} \;|\neq o' \succeq o$ or $\mathcal{N} \;|\neq o \succeq o'$ is $O(n)$ over the number n of attributes involved in the CP-Net $\mathcal{N}$ (Boutilier et al., 2004). On the other hand, the dominance problem is polynomial (when the graph verifies some conditions) and NP-complete in general. For a deeper discussion on these topics, see (Boutilier et al, 2004). In (Boutilier et al. 1997) it was shown that the dominance problem is intrinsically related to the problem of finding optimal outcomes satisfying a set of given constraints (the constraint-based preferential optimization problem in CP-Nets).

2.2 Tradeoffs-enhanced conditional preference networks (TCP-Nets)

In preference elicitation with CP-Nets the user describes how her preference over the values of an attribute depends on the values of other attributes. CP-Nets are able to specify a class of intuitive and useful preference statements of the form: "*I prefer the value a_0 for attribute A given that B = b and C = c*". However, there are other intuitive and important preference statements which cannot be represented by a CP-Net. These statements have the form: "*It is more important to me that the value of attribute A be better than the value of attribute B be better*". For instance, I could say that when choosing a movie, a most preferred genre is more important than a most preferred director. So, when comparing the films $f_1 = (c, w)$ and $f_2 = (t, n)$ in Example 2, I would prefer f_2 to f_1. Notice that the CP-Net $\mathcal{N}$ given in Example 2 is not able to infer $f_2 \succeq f_1$ nor $f_1 \succeq f_2$. Another kind of intuitive statements which cannot be represented by a CP-Net has the form: "*Given that C = c, a better assignement of attribute A is more important to me than a better assignement of attribute B*". For instance, I could say that when choosing a movie produced in the 50's, a most preferred genre is more important than a most preferred director. However, for movies produced during the 60's, directors play a more important role in my decision than the movie genre.

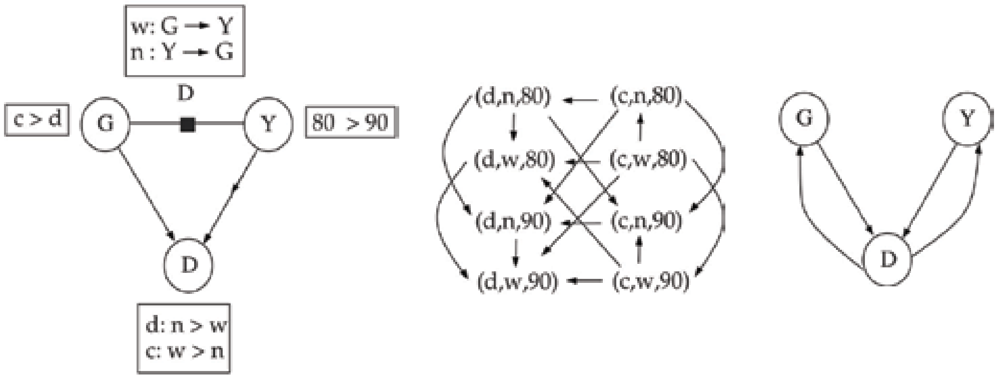
A CP-Net is able to specify only one kind of relationship between attributes, the *conditional preference dependence* relationship. In this section we consider an extension of the CP-Net formalism allowing two other kind of relationships between attributes: *relative importance* (atribute A is more important than attribute B in my decision) and *conditional relative importance* (attribute A is more important than attribute B in my decision given that the value for attribute C is c_0). This enhanced model, introduced in (Brafman et al., 2006a), is called *Tradeoffs-enhanced Conditional Preference Network (TCP-Nets)*.

Like CP-Nets, TCP-Nets are annotaded graphs where nodes are attributes. Unlike CP-Nets, TCP-Nets have three types of edges. The first one corresponds to CP-Nets edges, indicating conditional preference between attributes. The second edge type (directed) capture *relative importance* of attribute X over attribute Y. More precisely, let X and Y be two attributes mutually preferenctially independent given $Z = V - \{X, Y\}$, that is, for every fixed instantiation of the attributes in Z, the ranking of X values is independent of the value of Y. We say that X is more important than Y, denoted $X \rhd Y$, if for every instantiation $\mathbf{z}$ of the attributes in Z and for every $x, x' \in \mathbf{dom}(X)$ such that $x \succ x'$ given $\mathbf{z}$, we have that $xy\mathbf{z} \succ x'y'\mathbf{z}$.

The third edge type (undirected) captures *conditional relative importance*. More precisely, let X and Y be a pair of attributes in V and let $Z \subseteq V - \{X, Y\}$. We say that X is more important than Y given $\mathbf{z} \in \mathbf{dom}(Z)$ iff for every $\mathbf{w} \in \mathbf{dom}(V - (\{X, Y\} \cup Z))$ we have: $xy\mathbf{zw} \succ x'y'\mathbf{zw}$ whenever $x \succ x'$ given $\mathbf{zw}$. We denote this relation by $X \rhd_{\mathbf{z}} Y$. Thus, an undirected edge of the third type between attributes X and Y, labelled with the set of attributes Z, means that $X \rhd_{\mathbf{z}} Y$ or $Y \rhd_{\mathbf{z}} X$, depending on the values of the attributes in Z. As in CP-Nets, each node X in a TCP-Net is annotaded with a CP table CPT(X). In addition, in TCP-Nets, each undirected edge labelled with Z between attributes X and Y is annotaded with a *conditional importance table (or CI table)* CIT(X, Y, Z), describing the relative importance of X and Y given the value of the corresponding importance-conditioning attributes Z.

Definition 3 (TCP-Net Preference Model) A TCP-Net $\mathcal{N}$ is a tuple (V, cp, i, ci, cpt, cit) where: (1) V is a set of attributes (the nodes of $\mathcal{N}$); (2) cp (conditional preference arcs) is a set of

directed arcs (X, Y), for $X, Y \in V$; (3) i (importance arcs) is a set of direct arcs (X, Y), for $X, Y \in V$ such that $X \rhd Y$; ci (conditional importance) is a set of undirected arcs $\{X, Y\}$ labelled with a set of attributes Z such that $X \rhd_{\mathbf{z}} Y$ or $Y \rhd_{\mathbf{z}} X$ depending on the assignement **z** of attributes in Z ; *cpt* associates a CP table CPT(X) to each node X of $\mathcal{N}$, where CPT(X) is a mapping from **dom**(Parents(X)) to strict partial orders over **dom**(X); *cit* associates a CI table CTI(X, Y, Z) indicating, for each instantiation $\mathbf{z} \in$ **dom**(Z), the relative importance of X and Y.
The following example illustrates the concept of TCP-Net as a formalism for specifying preferences.
Example 4 Let V = {Director (D), Genre (G), Year (Y) }, **dom**(D) = {Woody Allen (w), Nanni Moretti (n)}, **dom**(G) = {comedy (c), drama (d)}, **dom**(Y) = {80,90}. Let us suppose that I strictly prefer comedies to dramas but my preference about directors is conditioned to the film genre: When choosing dramas, I prefer Nanni Moretti's to Woody Allen's. However, for comedies, I prefer Woody Allen's to Nanni Moretti's. When choosing a film, the year of production is more important for my decision than the director. When choosing a Woody Allen's film, its genre is more important to me than its year of production. But for Nanni Moretti's films, the year of production is more important than the genre. These preference rules can be expressed by the TCP-Net depicted in Figure 2(a).



(a) (b) (c)

Fig. 2. (a) A TCP-net $\mathcal{N}$ (b) The partial ordering induced by $\mathcal{N}$ (c) The dependence graph

The semantics of a TCP-Net is defined in terms of the set of *strict partial orders* consistent with the constraints imposed by the preference and importance relations expressed by the graph edges, the CP and CI tables. As for CP-Nets, TCP-Nets semantics is based on the *ceteris paribus* semantics. We present here only the intuitive idea behind the semantics of a TCP-Net. For a more formal presentation, see (Brafman et al., 2006a). A strict partial order $\succ$ *satisfies* a TCP-Net $\mathcal{N}$ if the following intuitive conditions are verified: (1) in each CP table CPT(X), for every $\mathbf{z} \in$ **dom**(Z) (where Z = Parents(X)), two objects o and o' differing only on the attribute X and whose values on Z are given by **z**, are ordered by $\succ$ consistently with the ordering on the X values given in CPT(X); (2) if $X \rhd Y$, then any two objects o and o' differing only on the values of X and Y are ordered as $o \succ o'$ if the relationship $o[X] \succ o[X']$ appears in the CP table CPT(X) corresponding to the instantiation given by o[Parents(X)]; (3) in each CI table CPI(X, Y,Z), for every $\mathbf{z} \in \mathbf{dom}(Z)$ such that $X \rhd_{\mathbf{z}} Y$, any two objects o and o' differing only on the attributes X and Y and whose values on Z are given by $\mathbf{z}$ are ordered as $o \succ o'$ if the relationship $o[X] \succ o[X']$ appears in the CP table CPT(X) corresponding to the instantiation given by o[Parents(X)]. In Figure 2(b) one represents the relationships between objects which are entailed by the TCP-Net in Figure 2(a). The arrow from object o to object o' means that $o \succ o'$. The arrows resulting from transitivity are not showed in the figure. Notice that the arrow from film (d, n, 80) to film (c, n, 90) is inferred using the CI table CIT(G, Y,D) which imposes that, for Nanni Moretti's movies, the year of production is more important than the genre. So, as I prefer films produced in the 80's than films produced in the 90's, I prefer the first film to the second one, even if the first film is a drama and the second one is a comedy.

Definition 4 (Satisfiability of a TCP-Net) A TCP-Net $\mathcal{N}$ is satisfiable (or consistent) iff there is some strict partial order $\succ$ over $\mathcal{O}$ that satisfies it. Let o, $o' \in \mathcal{O}$. We say that $o \succ o'$ is implied (or inferred) by the TCP-Net $\mathcal{N}$ iff it is verified by all strict partial orders $\succ$ over $\mathcal{O}$ satisfying $\mathcal{N}$.

Satisfiability is a desired property for TCP-Nets since it is important to guarantee that the preference rules provided by the users do not lead to inconsistencies like "I prefer object o to object o' and object o' to object o". However, the definition of TCP-Net satisfiability does not provide a mechanism for testing TCP-Net consistency. Fortunately, for a large class of TCP-Nets consistency is guaranteed. This class of TCP-Nets is referred as *conditionally acyclic* and is defined as follows:

Definition 5 (Conditionally Acyclic TCP-Nets) Let $\mathcal{N}$ be a TCP-Net over the set of attributes V . We associate to $\mathcal{N}$ a graph $\mathcal{N}^*$, called *the dependency graph of* $\mathcal{N}$ in the following way: the nodes of $\mathcal{N}^*$ are the same as the nodes of $\mathcal{N}$. Each directed edge of $\mathcal{N}$ is a directed edge of $\mathcal{N}^*$. For each undirected edge $\{X, Y\}$ of $\mathcal{N}$, labelled by the set of attributes Z, we insert in $\mathcal{N}^*$ two directed edges (A,X) and (A, Y) for each attribute $A \in Z$. Besides, for each assignement $\mathbf{z} \in \mathbf{dom}(Z)$ of the attributes in Z, we insert a direct edge (X,Y) or (Y,X) depending on the information given in the CI table CIT(X,Y,Z) corresponding to the assignement $\mathbf{z}$. In that way, we are able to associate a set of directed graphs $\mathcal{G}(\mathcal{N})$ to the TCP-Net $\mathcal{N}$, one for each assignement of the attributes labelling the undirected edges of $\mathcal{N}$. We say that the TCP-Net is *conditionally acyclic* if each graph of $\mathcal{G}(\mathcal{N})$ is acyclic.

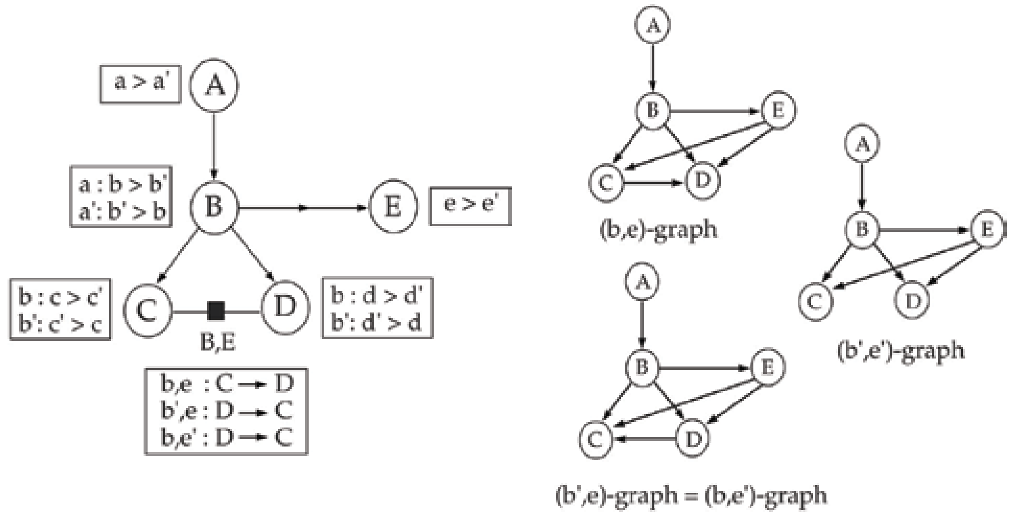
For instance, the dependence graph associated to the TCP-Net of Figure 2(a) is given in Figure 2(c). As we see, this TCP-Net is not conditionally acyclic, since the graph $\mathcal{N}^*$ is cyclic. Now, if we consider the TCP-Net depicted in Figure 3(a), it is easy to see that it is conditionally acyclic, since all graphs in $\mathcal{G}(\mathcal{N})$ (showed in Figure 3(b)) are acyclic.

For conditionally acyclic TCP-Nets we have the following result, whose proof can be found in (Brafman et al. 2006a).

Theorem 3 Every conditionally acyclic TCP-Net is satisfiable.

Discussion. (1) *Complexity*: Unfortunately, testing for conditionally acyclicity is not an easy task. This problem is shown to be coNP-hard in (Brafman et al. 2006a). (2) *Best Outcomes*:

One of the central properties of the CP-Net model is that, given an acyclic CP-Net $\mathcal{N}$ and a (possibly empty) partial instantiation **x** of some of its attributes, it is simple to determine a *best object* consistent with **x**. In the previous section, we presented the forward sweep procedure which produces the best object of an acyclic CP-Net. This procedure works also for conditionally acyclic TCP-Nets. The relative importance relations do not have any influence in the process of obtaining the optimal outcome. In order to obtain the best object, we simply consider the CP-Net part of the TCP-Net $\mathcal{N}$ (ignoring the *i*-edges and the *ci*-edges) and we apply the forward sweep procedure for the resulting CP-Net. This simple algorithm for finding the best outcome can be applied to all TCP-Nets for which the CP-Net part is acyclic. In particular, it is applicable for conditionally acyclic TCP-Nets. However, finding the best outcome associated to a TCP-Net $\mathcal{N}$ satisfying a set of hard constraints is not trivial. In (Brafman et al., 2006a), an algorithm (Search-TCP) is developed for producing the best outcomes associated to a conditionally acyclic TCP-Net $\mathcal{N}$ satisfying a set of hard constraints $\mathcal{C}$ on the attributes of $\mathcal{N}$.



(a) (b)

Fig. 3. (a) A conditionally acyclic TCP-net $\mathcal{N}$ (b) The set of acyclic graphs $\mathcal{G}(\mathcal{N})$

2.3 A logical framework for preferences over objects

In this section we present a third approach for preference elicitation and reasoning introduced in (Wilson, 2004). This approach is based on a logical framework and generalizes the CP-Nets and TCP-Nets approaches.

The Preference Language $\mathcal{L}$. The language $\mathcal{L}$ is constituted by statemets of the form φ: $u \rightarrow (X = x) > (X = x')$, where u is a formula of the form $(X_{i_1} = x_1) \wedge \ldots \wedge (X_{i_k} = x_k)$, with $X_{i_j} \in V - \{X\}$ and $x_j \in \mathbf{dom}(X_{i_j})$ for all $j \in \{1, \ldots, k\}$ and $x, x' \in \mathbf{dom}(X)$. We call such statements *conditional preference rules* or *cp-rules* for short. The formula u is called the *condition* of the cp-rule φ. The set of attributes appearing in u is denoted by *Attr(u)*. If φ is the statement $u \rightarrow$ (X

$= x) > (X = x')$ then sometimes we denote u by u_φ, X by X_φ and x, x' by x_φ and x'_φ respectively. A *conditional preference theory* over V is a finite set of statements of $\mathcal{L}$.

Example 5 Let $V = \{G,D\}$ as in Example 2. Let φ_1 and φ_2 the following conditional preference rules:

$\varphi_1 : (G = c) \rightarrow (D = w) > (D = n)$,

$\varphi_2 : (D = n) \rightarrow (G = c) > (G = d)$.

Then $\Gamma = \{\varphi_1, \varphi_2\}$ is a conditional preference theory which expresses the first preference statement of Example 1.

A conditional preference statement $\varphi : u \rightarrow (X = x) > (X = x')$ induces a preference ordering on objects over V. Let $o = tyx$ and $o' = tyx'$ be objects over V, where y is an object over $Attr(u)$, t is an object over $V - (Attr(u) \cup \{X\})$. We say that o is *preferred* to o' according φ. The set of pairs of objects (o, o') where o is preferred to o' according to φ is denoted by φ^*. If Γ is a conditional preference theory, we denote by $>_\Gamma$ the transitive closure of the binary relation $\Gamma^* = \bigcup_{\varphi\in\Gamma} \varphi^*$.

Example 6 Let us compare the objects $o_1 = (c,w)$ and $o_2 = (d, n)$ according to Γ. We have that (c,w) is preferred to (c, n) according to φ_1. And (c, n) is preferred to (d, n) according to φ_2. Then, using transitivity, we conclude that (c,w) is preferred to (d, n), that is, $o_1 >_\Gamma o_2$.

Consistency Test. One important feature of preference conditional theories is that there is no need of eliciting a total order on the values of an attribute given each assignement to its parents, as in the CP-Net preference model. So, a conditional preference theory is a compact way of expressing preference: we can reason with any theory Γ specified by the user, provided this theory satisfies some properties which guarantee its *consistency*. Besides, the user can add new statements later on; because the logic used in the deduction system is monotonic, all previous deductions concerning preferences will hold.

Now, we present the concept of *consistency* for a preference conditional theory Γ. A *model* of Γ is a strict partial order (that is, a transitive and irreflexive relation) $>$ on objects $\mathcal{O}$ over V such that $>$ contains the induced ordering $>_\Gamma$. We say that Γ is *consistent* if there exists a model $>$ for Γ. It is easy to see that a theory Γ is consistent if and only if its induced relation $>_\Gamma$ is irreflexive, since $>_\Gamma$ is transitive by definition.

Example 7 The theory Γ presented in Example 5 is consistent. Indeed $>_\Gamma = \{(o_1, o_3), (o_3, o_4), (o_1, o_4)\}$ is a strict partial order over the set of objects $\mathcal{O} = \{o_1, o_2, o_3, o_4\}$, where $o_1 = (c,w)$, $o_2 = (d,w)$, $o_3 = (c, n)$, $o_4 = (d, n)$. Note that $(o_1, o_3) \in \varphi_1^*$, $(o_3, o_4) \in \varphi_2^*$ and (o_1, o_4) is inferred by transivity. However, the theory $\Gamma' = \Gamma \cup \{\varphi_3, \varphi_4\}$ where: $\varphi_3 : (G = d) \rightarrow (D = n) > (D = w)$ and $\varphi_4 : (D = w) \rightarrow (G = d) > (G = c)$, is not consistent. Indeed, $(o_4, o_2) \in \varphi_3^*$ and $(o_2, o_1) \in \varphi_4^*$. So, $o_1 >_{\Gamma'} o_1$, since $(o_4, o_1) \in >_{\Gamma'}$ and $(o_1, o_4) \in >_{\Gamma'}$, which proves that $>_{\Gamma'}$ is not irreflexive.

We associate to each preference conditional theory Γ a graph $G(\Gamma)$ defined as follows: the nodes of $G(\Gamma)$ are the attributes appearing in the rules of Γ and the set of edges is given by $\{(Y,X_\varphi) : Y \in U_\varphi\}$, where U_φ denotes the set of attributes appearing in the condition u_φ. The preference conditional theory Γ is *acyclic* if its graph $G(\Gamma)$ is acyclic.

As we will see in Theorem 4, in order to ensure consistency for acyclic theories it will be sufficient to ensure *local consistency*. More precisely : Let o be a fixed object over V and X be an attribute in V . Let $x, x' \in \mathbf{dom}(X)$. We say that (x, x') is *validated* by o if there exists a statement $(\varphi : u_\varphi \rightarrow X = x > X = x') \in \Gamma$ such that o satisfies the formula u_φ (the conditions of φ). We define the relation $>_o^X$ on $\mathbf{dom}(X)$ as the transitive closure of the set of all pairs (x, x') validated by o. We say that the preference theory Γ is *locally consistent* if for all objects o and all attributes X, the relation $>_o^X$ is irreflexive.

Example 8 Let us consider the situation of Example 7 except that the set of attributes V is augmented with a third attribute Y (year of production), so $V = \{G,D, Y\}$. Let us consider the preference theory $\Gamma_1 = \{\varphi_1, \varphi_5\}$, where $\varphi_5 : (Y = 1990) \rightarrow (D = n) > (D = w)$. Let $o = (c,w, 1990)$ and let us fix the attribute D. Then $w >_o^D n$ since (w, n) is validated by o, if we consider the statement φ_1. But (n,w) is also validated by o, if we consider the statement φ_5. Thus, Γ_1 is not locally consistent.

The following theorem gives necessary and sufficient conditions for ensuring consistency of a preference theory Γ.

Theorem 4 Let Γ be a conditional preference theory. Then, we have : (1) If Γ is consistent then Γ is local consistent. (2) If Γ is local consistent and acyclic then Γ is consistent. (3) If all the attributes in V are binary, local consistency can be determined in time proportional to $|\Gamma|^2 \times |V|$.

The theory Γ_1 presented in Example 8 is not locally consistent, so it is not consistent by Theorem 4. Notice that its graph $G(\Gamma_1) = \{(G,D),(Y,D)\}$ is acyclic. On the other hand, the theory Γ given in Example 5 is consistent but its graph $G(\Gamma) = \{(G,D),(D,G)\}$ is cyclic. By Theorem 4 we can conclude that it is local consistent. This is an example of a local consistent theory whose graph is cyclic.

Finding optimal outcomes. Let Γ be a preference conditional theory over a set of attributs V. Given an object o over $V' \subseteq V$, we say that a value $x_i \in \mathbf{dom}(X_i)$ is undominated given the object o if there is no statement $u \rightarrow (X_i = x) > (X_i = x_i)$ in Γ, such that o satisfies u. The algorithm for finding optimal objects with respect to a locally consistent preference theory Γ with acyclic $G(\Gamma)$ works as follows: (1) enumerate the attributes of V in such a way that the ordering $\langle X_1, ...,X_n\rangle$ is compatible with the graph $G(\Gamma)$ (that is, if $i > j$ then there is no path going from X_j to X_i in $G(\Gamma)$). (2) For each $i \in \{1, ..., n\}$ let $\alpha(X_i) = x$, where $x \in \mathbf{dom}(X_i)$ and x is undominated with respect to the object $o = (\alpha(X_1), ..., \alpha(X_{i-1}))$. Local consistency of Γ ensures that such x always exists.

Example 9 Let us consider $V = \{G, D, Y\}$ as in Example 8. Let us consider the preference theory $\Gamma_1 = \{\varphi_1, \varphi_6\}$, where $\varphi_6 : (G = d) \rightarrow (Y = 1990) > (Y = 2000)$. We have $G(\Gamma_1) = \{(G,D), (G, Y)\}$. Then, the ordering $\langle G, D, Y\rangle$ is compatible with $G(\Gamma_1)$. We choose $\alpha(G) = c$. For $\alpha(D)$, the only undominated value given (c) is w. For $\alpha(Y)$, both values 1990 and 2000 are undominated given (c,w). So, a best object is $o_1 = (c,w, 1990)$. Another one is $o_2 = (c,w, 2000)$. By choosing $\alpha(G) = d$, we also get $o_3 = (d,w, 1990)$ and $o_4 = (d, n, 1990)$ as best objects.

3. Preferences over sets of objects

For the time being, we have been interested in formalisms allowing to eliciting and reasoning with preferences over objects. We have introduced some important frameworks for specifying user's preferences in a compact way, besides discussing important issues related to this topic, such as decidability and complexity of the problems of finding the best objects, dominance and ordering queries and introduction of hard constraints. In this section we tackle these issues in a broader context, by considering a simple framework for dealing with preferences over *sets of objects*. This problem arises naturally in the context of our running example. Let us suppose the task of creating a program for a film festival. Here, the crucial task is not to obtain user's preferences about movies considered individually, but about several possible *sets* of movies. Thus, the user has to be able to specify her preferences about a group of films, taking into account aspects like genre diversity, genre adaptability (for instance, a user may not be interested in programs containing both comedies and dramas), etc. In such situation, we would like to be able to determine the preferred characteristics which must be satisfied by a *group of objects*, and then to be able to select from a set of objects the best subset satisfying these preference rules. A simple way to treat the problem of finding the best subset of objects is to produce a set containing the k best elements according to a set of preference rules on individual objects. This naive solution is not suitable since the attractiveness of particular objects does not imply that these same objects put together would constitute an atractive set. If one of the requirements for a "good" set is the diversity of its elements, putting together a set of good objects would not necessarily produce the required diversity. Several recent works on preference modelling in AI have focused on eliciting and reasoning with preference over sets of objects, from a quantitative and a qualitative perspectives. In (des Jardins & Wagstaff, 2005) for instance, it was proposed a formalism to deal with preferences over sets of objects supporting the notions of *diversity* and *depth*. These concepts allows expressing preferences in a quantitative way, by *measuring* in some sort the degree of diversity and depth of a preferred set of objects. Since in this chapter we are focusing on formalisms based on a qualitative perspective, we will describe here the very simple and elegant approach of (Brafman et al., 2006b). This approach allows the user to specify a broad class of interesting properties about sets of objects. And surprisingly, such set-preference statements can be naturally transformed into conditional preference statements over attributed objects.

The Specification Language. Most properties of sets of objects which are important for users when specifying their preferences take the forms: (1) "*at least one object in the set satisfies* $C = c$ *and* $D = d$ *or* $A = a$; (2) *the number of objects satisfying* $C = c$ *is* 2. Let $\mathcal{L}$ the propositional language where the propositions are of the form $X = x$, where X is an attribute in the set V of attributes and $x \in \mathbf{dom}(X)$. An object $o \in \mathcal{O}$ *satisfies* $X = x$ if $o[X] = x$. This notion of *satisfaction* is extended to formulae $\varphi \in \mathcal{L}$ as usually in Propositional Logic. It is denoted by $o \mid= \varphi$. Now we consider the following class of properties $\mathcal{C}$ over sets of objects : $\langle |\varphi| \ \theta n \rangle$, where $\varphi \in \mathcal{L}$, $\theta \in \{=, \leq, \geq, >, <\}$, $n \in \mathbb{N}$. Using such statements, the user is able to express the properties about sets of objects which may affect her preferences. These properties refers to the number of objects in the selected subset O of objects verifying some constraints. The following example illustrates these properties:

Example 10 Let us consider the situation depicted in Example 2, but with an extra attribute standing for the film title. So, $V = \{G,D, T\}$. Let us suppose the following properties which affect user's preferences about a film program: the fact the it contains at most two Woody Allen's comedies, at least two Hitchcock's thrillers and no dramas. This can be specified by the following set-preference properties:

P_1: $\langle |G = c \wedge D = w| \leq 2\rangle$

P_2: $\langle |G = t \wedge D = h| \geq 2\rangle$ P_3: $\langle |G = d| = 0\rangle$

Let us consider the following set of films: $O = \{(c,w, t_1), (c,w, t_2), (c,w, t_3), (d, n, t_4), (t, h, t_5), (t, h, t_6)\}$. For this set we have $P_1(O)$ = false, $P_2(O)$ = true and $P_3(O)$ = false.

Now, let us consider a set of properties $\mathcal{P} = \{P_1, ..., P_n\} \in \mathcal{C}$. Each property P_i can be treated as an attribute taking values in the set {true, false} (**dom**(P_i) = {true, false}). Each subset of objects $O \subset \mathcal{O}$ corresponds to an "object" in $\mathcal{V}$ = **dom**(P_1) × .... × **dom**(P_n). That is, abstractly, each subset O can be viewed as a vector of truth-values (an object). Moreover, any preference order over objects in $\mathcal{V}$ implicitly induces a preference order over sets of objects of $\mathcal{O}$. So, in order to specify preferences over sets of objects, the user must simply specify (1) which are the properties about sets that affects her preferences and (2) her specific preference rules involving the validity of these properties. After such specifications, the problem of extracting a preference ordering over sets of objects satisfying the user's requirements is reduced to the problem of extracting a preference ordering over objects. Thus, we can use one of the formalisms introduced in the previous section for reasoning with preferences over objects in order to infer a preference ranking over sets of objects. The following example illustrates this idea.

Example 11 Let us consider the situation of our Example 10. Let us suppose the user specifies the following preference statements: (1) She prefers programs containing at most two Woody Allen's comedies; (2) For programs containing more than two Woody Allen's comedies she prefers a program containing at least one drama; (2) For programs containing no dramas he prefers a program containing at least two Hitchcock's thrillers. These preference statements can be represented by the TCP-Net depicted in Figure 4.

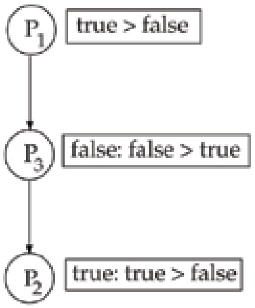


Fig. 4. A TCP-Net representing set-preference statements

4. Preferences over sequences of objects

In this section, we present our formalism allowing to specify compact preference statements provided by the users. First, we will formalize the notion of *temporal conditions* used for ranking sequences of objects. By viewing each object in a sequence σ as a state, we propose to use the formalism of Propositional Linear Temporal Logic (PTL) to capture the desired properties of sequence of objects, which we call *temporal conditions*. After formalizing our temporal conditions, we introduce the language **TPref** for expressing conditional preferences over sequences of objects. Preference statements in **TPref** use temporal conditions in their formulation.

4.1 Temporal conditions

The language we use for expressing temporal condition is basicly the Propositional Temporal Logic (PTL), adapted to our context. In PTL, the basic formulae are propositional variables $p_1, ..., p_n$. In our case, basic formulae or *propositions* are of the form $X = a$ where $X \in V$ and $a \in$ **dom(X)**. In order to emphasize the fact that our language assume a particular basic formula format, we will call it STL (for Simple Temporal Logic) instead of PTL. We stress however that both logics are essentially the same.

Definition 6 (The language STL for temporal conditions) The STL formulae are defined as follows: (1) **true** and **false** are STL formulae. (2) if P is a proposition then P is a STL formula. (3) if F and G are STL formulae then $F \wedge G$, $F \vee G$ and $\neg F$ are STL formulae. (4) if F and G are STL formulae then F **Until** G and F **Since** G are STL formulae. A *temporal condition* is a STL formula. If F is a temporal condition, we denote by Attr(F) the set of attributes appearing in F.

Next, we present the semantics of temporal conditions. Temporal conditions are evaluated over *sequences* of objects. A *sequence of objects* of $\mathcal{O}$ is a structure consisting of a set of objects $\{o_1, o_2, ..., o_k\}$ with an (temporal) ordering $o_1 < o_2 < ... < o_k$, telling us that o_i comes before o_{i+1}. We denote this structure simply by $\sigma = \langle o_1, o_2, ..., o_k\rangle$. If $\sigma = \langle o_1, ..., o_k\rangle$ then k is called the *length* of σ and is denoted by $|\sigma|$. We denote by $Seq(\mathcal{O})$ the set of sequences of objects in $\mathcal{O}$ and by $Seq_n(\mathcal{O})$ the set of sequences of length n in $Seq(\mathcal{O})$.

Definition 7 (STL Semantics) The notion of *satisfaction* of a **STL** formula by a sequence of objects $\sigma = \langle o_1, ..., o_k\rangle$ at a state $i \in \{1, ..., k\}$ (denoted by $(\sigma, i) \models F$) is inductively defined as follows: (1) $(\sigma, i) \models (X = a)$ iff $o_i[X] = a$; (2) $(\sigma, i) \models F \wedge G$ iff $(\sigma, i) \models F$ and $(\sigma, i) \models G$; (3) $(\sigma, i) \models F \vee G$ iff $(\sigma, i) \models F$ or $(\sigma, i) \models G$; (4) $(\sigma, i) \models \neg F$ iff $(\sigma, i) \not\models F$;
(5) $(\sigma, i) \models F$ **Until** G iff there exists j such that $i < j \leq |\sigma|$ and $(\sigma, j) \models G$ and for all k such that $i < k < j$ we have $(\sigma, k) \models F$.
(6) $(\sigma, i) \models F$ **Since** G iff there exists j such that $1 \leq j \leq i$ and $(\sigma, j) \models G$ and for all k such that $j < k < i$ we have $(\sigma, k) \models F$.
We say that σ *satisfies* a **STL** formula F (denoted by $\sigma \models F$) if $(\sigma, k) \models F$, where $k = |\sigma|$. We say that F is *satisfiable* if there exists $\sigma \in Seq(\mathcal{O})$ such that $\sigma \models F$. The formula **true** (resp. **false**) is satisfied by any sequence (resp. by no sequence) $\sigma \in Seq(\mathcal{O})$. We say that two **STL** formulae F,G are *equivalent* iff for every sequence σ, $\sigma \models F$ iff $\sigma \models G$. We say that F,G are *globally equivalent* (g-equivalent) iff for every sequence σ, $(\sigma, i) \models F$ iff $(\sigma, i) \models G$, for all $i \in \{1, ..., |\sigma|\}$.

Derived Formulae:
Prev F = **false Since** F ("in the previous state F"); **Next** F = **false Until** F ("in the next state F"); **First** = $\neg$ **Prev true** (*"I am at the first state"*); **Last** = $\neg$ **Next true** (*"I am at the last state"*).; $\blacklozenge F$ = **true Since** F (*"Sometimes in the past F"*); $\Diamond F$ = **true Until** F (*"Sometimes in the future F"*); $\blacksquare F = \neg\blacklozenge\neg F$ (*"Always in the past F"*); $\square F = \neg\Diamond\neg F$ (meaning *"Always in the future F"*)

A very important property verified by PTL formulae (and consequently, by STL formula) is the *separability property*: it says that every PTL formula is g-equivalent to a boolean combination of *pure past*, *pure future* and *pure present* formulae. Let us define these kind of formulae:

Definition 8 (Present, Past and Future Formulae) A *pure present* formula is inductively defined by the following rules: (1) a proposition $X = a$ is a pure present formula. (2) a boolean combination of pure present formulae is a pure present formula. A *pure past* formula (resp. a *pure future* formula) is inductively defined as follows: (1) if F and G are pure present formulae then F **Since** G (resp. F **Until** G) are pure past formula (resp. a pure future formula). (2) If F and G are pure past formulae (resp. pure future formulae) then F **Since** G (resp. F **Until** G) is a pure past formula (resp. a pure future formula). (3) a boolean combination of pure past formulae (resp. pure future formulae) is a pure past formula (resp. a pure future formula). We say that a formula F is *separated* if F is of the form $F_1 \vee ... \vee F_n$, with each F_i of the form $F_i^0 \wedge F_i^+ \wedge F_i^-$, where F_i^0 is pure present formula, F_i^+ is a pure future formula and F_i^- is a pure past formula.

From a semantic point of view, the pure present, pure past and pure future formulae verifies the following properties which are easily proved by induction on the formulae construction.

Proposition 1 Let F be a STL formula.

- F is a pure present formula iff for all $\sigma = \langle o_1, ..., o_{i-1}, o_i, o_{i+1}, ..., o_k \rangle \in Seq_k(\mathcal{O})$ we have: $(\sigma, i) \models \varphi$ iff $(\sigma', i) \models \varphi$ for any sequence $\sigma' = \langle o_1, ..., o_{i-1}, o'_i, o_{i+1}, ..., o_k \rangle \in Seq_k(\mathcal{O})$ which differ from σ only at state i.
- F is pure past formula iff for all $\sigma = \langle o_1, ..., o_{i-1}, o_i, ..., o_k \rangle \in Seq_k(\mathcal{O})$ we have: $(\sigma, i) \models \varphi$ iff $(\sigma', i) \models \varphi$ for any sequence $\sigma' = \langle o_1, ..., o_{i-1}, o'_i, ..., o'_k \rangle$.
- F is pure future formula iff for all $\sigma = \langle o_1, ..., o_i, o_{i+1}..., o_k \rangle \in Seq_k(\mathcal{O})$ we have: $(\sigma, i) \models \varphi$ iff $(\sigma', i) \models \varphi$ for any sequence $\sigma' = \langle o'_1, ..., o'_i, o_{i+1}, ..., o_k \rangle$.

Intuitively, pure past formulae are not "aware" of what is happening in the current state or in future states. Pure present formulae are not "aware" of what has happened in the past states or of what is going to happen in future states. And pure future formulae are not "aware" of what is happening in the current state or has happened in past states.

Theorem 5 (Separation Theorem (Gabbay, 1989)) Let F be a STL formula. Then F is g-equivalent to a separated formula.

For instance, $\lozenge((X = a) \wedge \blacksquare(Y = b))$ is not separated but is equivalent to the separated formula $\blacksquare(Y = b) \wedge (Y = b) \wedge ((Y = b)$ **Until** $(X = a))$. The property of separation of propositional temporal formulae is not trivial. In fact, separation is closely related to the expressivity power of a temporal language. For details on this important subject see (Gabbay, 1989). For a discussion about open problems concerning the complexity of separating a formula into its past, future and present components see (Hodkinson & Reynolds, 2005).

4.2 A temporal preference language

Now, we introduce the specification language for our temporal preference model. A temporal preference will be characterized by a set of *temporal conditional preference rules* that we formally define next.

Definition 9 (Temporal Conditional Preference Rule) A*temporal conditional preference rule* (or *tcp-rule*) is an expression of the form: $\varphi : F \rightarrow (X = x > X = x')$ where $X \in V$, $x, x' \in$ **dom**(X) and F is a STL separated formula. A *simple* tcp-rule is a tcp-rule where the temporal condition contains a unique disjunct. It is easy to see that a tcp-rule is equivalent to a set of *simple* tcp-rules.

Definition 10 (Temporal Conditional Preference Theory) A *Temporal Conditional Preference-Theory* is a finite set Φ of simple tcp-rules $F \rightarrow (X = x) > (X = x')$, where $X \notin \text{Attr}(F_0)$. In what follows, sometimes it will be useful to use the following notation for the elements appearing in a tcp-rule φ: $F^-_\varphi \wedge F^0_\varphi \wedge F^+_\varphi$ denotes its temporal condition and $(X_\varphi = x_\varphi) > (X_\varphi = x'_\varphi)$ denotes the expression appearing in its right side.

Example 12 Let us consider the situation of our film festival program presented in Example 1. The statements are expressed by the following tcp-rules:

1. $\varphi_1 : (G = c) \rightarrow (D = w) > (D = c)$
2. $\varphi_2 : (G = d) \rightarrow (D = n) > (D = w)$. Here, the conditions in the tcp-rules are pure present formulae.
3. φ_3 : **First** $\rightarrow (G = c) > (G = d)$. Here, the condition in the tcp-rule is a pure past formula since **First** ≡ **Prev False**.
4. φ_4 : **Prev**$(G = c) \rightarrow (G = d) > (G = c)$
5. φ_5 : **Prev**$((G = d) \wedge (D = w)) \rightarrow (G = c) > (G = d)$
6. φ_6 : **Prev**$((G = d) \wedge (D = n)) \rightarrow (G = d) > (G = c)$. Here, the conditions in the tcp-rule are pure past formulae.
7. φ_7: $(\Diamond(G = d) \wedge \blacklozenge(G = c)) \rightarrow (G = c) > (G = d))$. Here, the conditions in the tcp-rules are separated formulae of the form $F^- \wedge F^+$ (with pure past and pure future components only).

The ordering induced by a Temporal Preference Theory. First of all we will show how two sequences in $Seq(\mathcal{O})$, differing at one single position i, can be compared via a temporal preference theory. Afterwards, we show how two sequences in $Seq(\mathcal{O})$, differing in k positions $i_1, ..., i_k$ can be compared.

Definition 11 (Sequences differing at one single position) Let φ be a tcp-rule. Let R_φ be the relation over $Seq_n(\mathcal{O})$ defined as follows: if $\sigma = \langle o_1, \ldots, o_n \rangle$ and $\sigma' = \langle o'_1, \ldots, o'_n \rangle$ then $\sigma R_\varphi \sigma'$ iff there exists $j \in \{1, \ldots, n\}$ such that: (1) $o_j \neq o'_j$ and $o_i = o'_i$ for every $i \in \{1, \ldots, n\} \backslash \{j\}$; (2) $(\sigma, j) \mid= F_\varphi$ and $(\sigma', j) \mid= F_\varphi$; (3) $o_j [X_\varphi] = x_\varphi$ and $o'_j [X_\varphi] = x'_\varphi$; (4) For every $Y \in V \setminus \{X_\varphi\}$, $o_j [Y] = o'_j [Y]$. If such position j exists, it is unique and denoted by $\delta(\sigma, \sigma')$.

Thus, two sequences of the same size can be compared via R_φ only if they differ at one single position. Roughly speaking, in order to compare two sequences differing at $k > 1$ positions, via a temporal conditional preference theory Φ, we will consider the union of R_φ, for $\varphi \in \Phi$ and the transitive closure of this union. More precisely: Given a set Φ of tcp-rules, we denote by R_Φ the set $\bigcup_{\varphi \in \Phi} R_\varphi$ and by $>_\Phi$ the transitive closure of R_Φ. We say that σ is *preferred* to σ'

w.r.t. the theory Φ if $\sigma >_\Phi \sigma'$. Lemma 1 below gives a necessary and sufficient condition in order to a sequence σ be preferred to a sequence σ' w.r.t. Φ. Before stating this result, we need the following definition:

Definition 12 (Improving Flipping Sequence (IFS)) Let σ and σ' be two sequences of length n. We say that there exists an *Improving Flipping Sequence (IFS)* from σ to σ' w.r.t Φ if there exists a set of sequences $\{\sigma_1, \ldots, \sigma_{p+1}\}$ and a set of tcp-rules $\{\varphi_1, \ldots, \varphi_p\}$ in Φ such that $\sigma_1 = \sigma$, $\sigma_{p+1} = \sigma'$ and $\sigma_k\ R_{\varphi k}\ \sigma_{k+1}$ for every $k \in \{1, \ldots, p\}$.

Lemma 1 Let Φ be a set of tcp-rules. Let σ and σ' be two sequences of length n. Then $\sigma >_\Phi \sigma'$ iff there exists an IFS from σ to σ' w.r.t. Φ.

Example 13 Let us consider the theory $\Phi = \{\varphi_1, ..., \varphi_7\}$ of Example 12 and the following sequences: $\sigma_1 = \langle(c, n), (d,w)\rangle$, $\sigma_2 = \langle(d, n), (d,w)\rangle$ and $\sigma_3 = \langle(d, n), (c,w)\rangle$. Note that $\delta(\sigma_1, \sigma_2) = 1$ and $\delta(\sigma_2, \sigma_3) = 2$. So, σ_1 and σ_3 differ in two positions, 1 and 2. We have $\sigma_1\ R_{\varphi 7}\ \sigma_2$ and $\sigma_2\ R_{\varphi 6}\ \sigma_3$. Then there exists an IFS from σ_1 to σ_3, and so $\sigma_1 >_\Phi \sigma_3$.

As we see, a temporal conditional preference theory Φ is a compact way of expressing preference between sequences of objects: we can reason with any theory Φ the user gives us, provided this theory is *consistent*. More precisely:

Definition 13 (Consistency) Let Φ be a temporal preference theory. We say that Φ is *consistent* iff $>_\Phi$ is irreflexive, that is, $>_\Phi$ is a partial order over $Seq_n(\mathcal{O})$, for all $n > 0$ (remind that, by definition, $>_\Phi$ is transitive; and that transitivity and irreflexivity imply anti-symmetry).

4.3 Consistency test

The main purpose of this section is to give necessary and sufficient conditions for a temporal conditional preference theory Φ to be consistent. In this paper, we only give necessary and sufficient conditions when tcp-rules in Φ use only conjunctions of pure past and pure present formulae of STL, i.e. for all $\varphi \in \Phi$, $F_\varphi = F_\varphi^- \wedge F_\varphi^0$. In the following, we denote by **TPref*** the set of all tcp-rules of this form.

A Method for Testing Consistency. We will show (Theorem 6) that testing the consistency of a temporal conditional preference theory Φ reduces to test the consistency of a number $l(\Phi)$ of conditional preference theories over objects. Before proving this result, we need to introduce some notation first.

Let $\sigma = \langle o_1, \ldots, o_n\rangle$ be a sequence in $Seq_n(\mathcal{O})$ and o_{n+1} be an object in $\mathcal{O}$. In the following, we denote by *rlo* (for Remove Last Object) and *add* the operators defined by: $rlo(\sigma) = \langle o_1, \ldots, o_{n-1}\rangle$ and $add(\sigma, o_{n+1}) = \langle o_1, \ldots, o_n, o_{n+1}\rangle$. Let φ be a tcp-rule where $F_\varphi = F_\varphi^- \wedge F_\varphi^0 \wedge F_\varphi^+$. We denote by φ^0 the cp-rule defined by: $\varphi^0 : F_\varphi^0 \rightarrow (X_\varphi = x_\varphi) > (X_\varphi = x_\varphi)$. Given a tcp-theory Φ and a sequence $\sigma \in Seq(\mathcal{O})$, we define for every integer $j \in \{1, \ldots, |\sigma|\}$ the cp-theory $\Gamma_j(\Phi, \sigma)$ as follows:

$$\Gamma_j(\Phi, \sigma) = \{\varphi^0 \mid \varphi \in \Phi \wedge (\sigma, j) \models F^-_\varphi \wedge F^+_\varphi\}.$$

Intuitively, $\Gamma_j(\Phi, \sigma)$ is the set of the present components of the tcp-rules conditions whose past and future components are satisfied by σ at position *j*. Note that if σ and σ′ are two sequences in $Seq_n(\mathcal{O})$ such that $rlo(\sigma) = rlo(\sigma')$ then $\Gamma_n(\Phi,\sigma) = \Gamma_n(\Phi,\sigma')$. The following lemma gives a necessary condition for two sequences σ and σ′ satisfy $\sigma >_\Phi \sigma'$, where Φ is a theory in **TPref*** (without future components).

Lemma 2 Let Φ be a tcp-theory such that for every $\varphi \in \Phi$, $\varphi \in$ **TPref***. For every pair of sequences $\sigma = \langle o_1, \dots, o_{n+1}\rangle$ and $\sigma' = \langle o'_1, \dots, o'_{n+1}\rangle$ in $Seq_{n+1}(\mathcal{O})$ with $n > 0$, if $\sigma >_\Phi \sigma'$, then $rlo(\sigma) >_\Phi rlo(\sigma')$, or $rlo(\sigma) = rlo(\sigma')$ and $o_{n+1} >_\Gamma o'_{n+1}$ where $\Gamma = \Gamma_{n+1}(\Phi,\sigma) = \Gamma_{n+1}(\Phi,\sigma')$.

Proof. Let $\sigma = \langle o_1, \dots, o_{n+1}\rangle$ and $\sigma' = \langle o'_1, \dots, o'_{n+1}\rangle$ be two sequences such that $\sigma >_\Phi \sigma'$. If $\sigma >_\Phi \sigma$, it means that there exists an IFS from σ to σ′ w.r.t. Φ. Thus, there exists a set of sequences $\{\tau_1, \dots, \tau_{p+1}\}$ in $Seq_{n+1}(\mathcal{O})$ and a set of tcp-rules $\{\varphi_1, \dots, \varphi_p\}$ such that $\tau_1 = \sigma$, $\tau_{p+1} = \sigma'$ and for every $k \in \{1, \dots, p\}$, $\tau_k\, R_{\varphi k}\, \tau_{k+1}$. For every $k \in \{1, \dots, p+1\}$, let τ'_k be the sequence in $Seq_n(\mathcal{O})$ defined by $\tau'_k = rlo(\tau_k)$. It can be easily seen that for every $k \in \{1, \dots, p\}$, we have:

- $\tau'_k = \tau'_{k+1}$ if $\tau_k[n + 1] \neq \tau_{k+1}[n + 1]$, or
- $\tau'_k \neq \tau'_{k+1}$ if $\tau_k[n+1] = \tau_{k+1}[n+1]$. In that case, we have $j = \delta(\tau_k, \tau_{k+1}) = \delta(\tau'_k, \tau'_{k+1}) < n+1$. Therefore, since $\varphi \in$ **TPref***, $(\tau_k, j) \models F_{\varphi k}$ and $(\tau_{k+1}, j) \models F_{\varphi k}$ implies that $(\tau'_k, j) \models F_{\varphi k}$ and $(\tau'_{k+1}, j) \models F_{\varphi k}$. Since $\tau_k\, R_{\varphi k}\, \tau_{k+1}$, it follows that we also have $\tau'_k\, R_{\varphi k}\, \tau'_{k+1}$.

We now have to distinguish two cases:

1. Assume that there exists an integer $k \in \{1, \dots, p\}$ such that $\tau'_k \neq \tau'_{k+1}$. In that case, since $\tau'_1 = rlo(\sigma), \tau'_{p+1} = rlo(\sigma')$, we have shown that there exists an IFS from σ to σ′ w.r.t. Φ. It shows that $rlo(\sigma) >_\Phi rlo(\sigma')$.
2. Assume now that for every $k \in \{1, \dots, p\}$, we have $\tau'_k = \tau'_{k+1}$. It means that for every $k \in \{1, \dots, p+1\}$, $rlo(\tau_k) = rlo(\sigma) = rlo(\sigma')$. Moreover, since $\tau_k\, R_{\varphi k}\, \tau_{k+1}$ and $\delta(\tau_k, \tau_{k+1}) = n+1$, we have $(\tau_k, n + 1) \models F_{\varphi k}$. It follows that $(\tau_k, n + 1) \models F^-_{\varphi k}$. Thus, since $rlo(\tau_k) = rlo(\sigma)$, we have $(\sigma, n + 1) \models F^-_{\varphi k}$ and $\varphi^0_k \in \Gamma_{n+1}(\Phi,\sigma)$. Now, it is easy to see that $(\tau_1[n + 1] = o_{n+1}) >_\Gamma (\tau_{p+1}[n + 1] = o'_{n+1})$ where $\Gamma = \Gamma_{n+1}(\Phi,\sigma)$, which completes the proof of Proposition 2. □

We now are ready to state the main result of this section. Its proof uses Lemma 2.

Theorem 6 Let Φ be a set of tcp-rules such that for every $\varphi \in \Phi$, $\varphi \in$ **TPref***. Φ is *consistent* iff for every sequence σ of length $k > 0$, $\Gamma_k(\Phi,\sigma)$ is consistent.

Proof. In order to prove that Φ is consistent, we have to show that $>_\Phi$ is irreflexive. First, we show that if for every sequence σ of length $k > 0$, $\Gamma_k(\Phi,\sigma)$ is consistent, then the relation $>_\Phi$ is irreflexive. We show this property by induction on the length of sequences.

Let $\sigma = \langle o\rangle$ be a sequence of length n = 1. If $\sigma >_\Phi \sigma$, it means that there exists an IFS from σ to σ, i.e. a set of sequences $\{\langle o_1\rangle, \dots, \langle o_{p+1}\rangle\}$ and a set of tcp-rules $\{\varphi_1, \dots, \varphi_p\}$ such that $o = o_1 = o_{p+1}$

and for every $k \in \{1, \dots, p\}$, $\langle o_k\rangle R_{\varphi k} \langle o_{k+1}\rangle$. By Definition 11, for every $k \in \{1, \dots, p\}$, we have $(\langle o_k\rangle, 1) \models F_{\varphi k}$. Thus, we have $(\langle o_k\rangle, 1) \models F^{-}_{\varphi k}$ and $\varphi^0_k \in \Gamma_1(\Phi,\langle o_k\rangle) = \Gamma_1(\Phi,\langle o\rangle)$ because $rlo(\langle o_k\rangle) = rlo(\langle o\rangle)$. Finally, for every $k \in \{1, \dots, p\}$, we have $o_k\ R_{\varphi^0_k}\ o_{k+1}$. It shows that $o >_\Gamma o$ with $\Gamma = \Gamma_1(\Phi,\langle o\rangle)$, which contradicts the hypothesis that $\Gamma_k(\Phi,\sigma_k)$ is consistent for every sequence σ_k of length $k > 0$ and proves that $>_\Phi$ is irreflexive on $Seq_1(\mathcal{O})$.

Assuming that $>_\Phi$ is irreflexive on $Seq_n(\mathcal{O})$, we now have to prove that $>_\Phi$ is a irreflexive on $Seq_{n+1}(\mathrm{O})$. Suppose that there exists a sequence $\sigma_{n+1} = \langle o_1, \dots, o_{n+1}\rangle$ in $Seq_{n+1}(\mathcal{O})$ such that $\sigma_{n+1} >_\Phi \sigma_{n+1}$. Using Proposition 2, we have to distinguish two cases:

1. If $rlo(\sigma_{n+1}) >_\Phi rlo(\sigma_{n+1})$, it shows that $>_\Phi$ is not irreflexive on $Seq_n(\mathcal{O})$, which contradicts the hypothesis.
2. If $rlo(\sigma_{n+1}) = rlo(\sigma_{n+1})$, then we have $o_{n+1} >_\Gamma o_{n+1}$ where $\Gamma = \Gamma_{n+1}(\Phi,\sigma_{n+1})$. It shows that $>_\Gamma$ is not irreflexive, which contradicts the hypothesis that $\Gamma_k(\Phi,\sigma_k)$ is consistent for every sequence σ_k of length $k > 0$.

So, we have proved by induction that if for every sequence σ of length $k > 0$, $\Gamma_k(\Phi,\sigma)$ is consistent, then $>_\Phi$ is a SPO on $Seq_n(\mathcal{O})$ for every integer $n \geq 1$.

We now prove that if $>_\Phi$ is a SPO, then $\Gamma_k(\Phi,\sigma)$ is consistent for every sequence σ of length $k > 0$. Assume that there exists a sequence σ of length k such that $\Gamma = \Gamma_k(\Phi,\sigma)$ is not consistent. It means that $>_\Phi$ is not irreflexive, i.e. that there exists an object o_{k+1} such that $o_{k+1} >_\Gamma o_{k+1}$. Let σ_{k+1} be the sequence defined by $\sigma_{k+1} = add(\sigma, o_{k+1})$. It is easy to see that $\sigma_{k+1} >_\Phi \sigma_{k+1}$, which contradicts the fact that $>_\Phi$ is irreflexive and completes the proof. □

Theorem 6 is not true when the tcp-rules in Φ contain past and future components, as we show in the following example:

Example 14 Let $\Phi = \{ \varphi_1, \varphi_2, \varphi_3, \varphi_4 \}$ be the set of tcp-rules defined by:

- $\varphi_1 : Next(G = d) \rightarrow (D = n) > (D = w)$
- $\varphi'_1 : Next(G = c) \rightarrow (D = w) > (D = n)$
- $\varphi_2 : Prev(D = n) \rightarrow (G = c) > (G = d)$
- $\varphi'_2 : Prev(D = w) \rightarrow (G = d) > (G = c)$

Since the STL formulae $\varphi_1 \wedge \varphi'_1$ and $\varphi_2 \wedge \varphi'_2$ cannot be satisfied, it is easy to see that for every sequence σ of length k, $\Gamma = \Gamma_k(\Phi, \sigma)$ is locally consistent. Moreover, for every sequence σ of length k, $G(\Gamma_k(\Phi,\sigma)) = (\{G,D\}, 3)$ is acyclic. Therefore, for every sequence σ of length k, $\Gamma = \Gamma_k(\Phi, \sigma)$ is consistent. We now show that Φ is not consistent, which does not contradict Theorem 6 since Φ uses past and future STL formulae in the conditions of the tcp-rules. Given the objects $o_1 = (c, n)$, $o_2 = (d,w)$, $o'_1 = (c,w)$ and $o'_2 = (c,w)$, consider the sequences $\sigma_1 = \langle o_1, o_2\rangle$, $\sigma_2 = \langle o'_1, o_2\rangle$, $\sigma_3 = \langle o'_1, o'_2\rangle$ and $\sigma_4 = \langle o_1, o'_2\rangle$. It is easy to verify the following:

- $\sigma_1\ R_{\varphi 1}\ \sigma_2$ since $(\sigma_1, 1) \models Next(G = d)$, $(\sigma_2, 1) \models Next(G = d)$, $o_1[D] = n$ and $o'_1[D] = w$.
- $\sigma_2\ R_{\varphi' 2}\ \sigma_3$ since $(\sigma_2, 2) \models Prev(D = w)$, $(\sigma_3, 2) \models Prev(D = w)$, $o_2[G] = d$ and $o'_2[G] = c$.

- $\sigma_3 R_{\varphi'1} \sigma_4$ since $(\sigma_3, 1) \models Next(G = c)$, $(\sigma_4, 1) \models Next(G = c)$, $o'_1[D] = w$ and $o_1[D] = n$.
- $\sigma_4 R_{\varphi 2} \sigma_1$ since $(\sigma_4, 2) \models Prev(D = n)$, $(\sigma_1, 2) \models Prev(D = n)$, $o'_2[G] = c$ and $o_2[G] = d$.

Thus, we have $\sigma_1 >_\Phi \sigma_1$, which shows that $>_\Phi$ is not consistent since it is not irreflexive.

Complexity Issues. In practice, the condition provided by Theorem 6 to test consistency is unfeasible, since it involves testing consistency of the non-temporal theories $\Gamma_k(\Phi, \sigma)$ for every sequence σ of length k. Fortunately, for some fragments of STL, we can find a very satisfatory bound for the size of the sequences σ which must be considered in the tests.

Theorem 7 Let $L(\blacklozenge, \lozenge)$ be the fragment of *STL* whose formulae satisfy the following conditions: (1) negation appear only in front of basic propositions; (2) the only temporal operators are $\blacklozenge$ and $\lozenge$. Let $F \in L(\blacklozenge,\lozenge)$ be satisfiable. Then there exists a sequence σ such that $|\sigma| \leq length(F)$ and such that σ satisfies F. The *length* of a formula F (denoted by *length(F)*) is the number of symbols appearing in F.

Proof. Let $\sigma = \langle o_1, ..., o_k\rangle$ be a sequence. A subsequence of σ is a sequence $\tau = \langle u_1, ..., u_m\rangle$ such that for all $i \in \{1, ...,m\}$ there exists $j_i \in \{1, ..., k\}$ such that $o_{i_j} = u_i$. We denote the fact that τ is a subsequence of σ by $\tau \preceq \sigma$.

Let $F \in STL$ and $\sigma = (o_1, ..., o_k)$ such that $(\sigma, i) \models F$. We will prove that there exists a subsequence $\tau \preceq \sigma$ such that (1) τ contains the object o_i, (2) $|\tau| \leq$ length(F) and (3) for all sequence σ' such that $\tau \preceq \sigma' \preceq \sigma$ we have $(\sigma', i) \models F$. Particularly, we can affirm that $(\tau, i) \models F$, since $\tau \preceq \tau \preceq \sigma$.

The proof is by induction on the structure of F.

- If F is atomic and $(\sigma, i) \models F$, then let $\tau =< o_i >$. We have that $\tau \preceq \sigma$, $|\tau| = 1 =$ length(F) and for all σ' such that $\tau \preceq \sigma' \preceq \sigma$ we have $(\sigma', i) \models F$, since σ' contains the object σ_i.
- If F is $\neg F_1$, where F_1 is an atomic formula, the proof is similar: we take $\tau =< o_i >$. In this case, $|\tau| = 1 <$ length(F) = 2.
- The cases where $F = G \vee H$ and $F = G \wedge H$ do not present any difficulty and we omit it here.
- If $F = \lozenge F_1$ and $(\sigma, i) \models F$. Then there exists $j > i$ such that $(\sigma, j) \models F_1$. By the induction hypothesis, we can affirm that there exist a subsequence $\tau \preceq \sigma$, such that τ contains the object o_j, $|\tau| \leq$ length(F_1), and for all σ' verifying $\tau \preceq \sigma' \preceq \sigma$ we have $(\sigma', j) \models F_1$. If $o_i \in \tau$ we define $\tau' = \tau$. Otherwise, τ' is obtained from τ by inserting the object o_i in it (in the same order as it appears in σ). Then, it is clear that $(\tau', i) \models \lozenge F_1$, since $(\tau', j) \models F_1$. Moreover, $\tau' \leq \tau + 1 =$ length(F). The proof is similar for $F = \blacklozenge F_1$.
- If $F =$ **Next** F_1 and $(\sigma, i) \models F$. Then $i < |\sigma|$ and $(\sigma, i+1) \models F_1$. By the induction hypothesis, we can affirm that there exist a subsequence $\tau \preceq \sigma$, such that τ contains the object o_{i+1}, $|\tau| \leq$ length(F_1), and for all σ' verifying $\tau \preceq \sigma' \preceq \sigma$ we have $(\sigma', i + 1) \models F_1$. If $o_i \in \tau$ we define $\tau' = \tau$. Otherwise, τ' is obtained from τ by inserting the object o_{i+1} in it (following the object o_i). Then, it is clear that $(\tau', i) \models$ **Next** F_1 since $(\tau', i + 1) \models F_1$. Moreover $\tau' \leq \tau + 1 =$ length(F). The proof is similar for $F =$ **Prev** F_1.

According to (Sistla & Clarke, 1985), the satifiability problem for STL is NP-complete for $L(\blacklozenge, \lozenge)$ and PSPACE-complete for the logic STL.

Proposition 2 Let Φ be a set of tcp-rules in **TPref*** such that the temporal conditions are formula of $L(\blacklozenge, \lozenge)$. Then Φ is consistent iff $\Gamma_k(\Phi, \sigma)$ is consistent for every sequence σ of length $\leq$ length(Φ), where length(Φ) = max{ length(φ) | $\varphi \in \Phi$ }.

Notice that if we place ourselves in a context where the universe of sequences is finite, then there is no need to restrict the conditions of the tcp-rules to be formulae in $L(\blacklozenge, \lozenge)$. As the whole universe of sequences is contained in $Seq_n(\mathcal{O})$, for some $n > 0$, then in order to test consistency of a tcp theory Φ, it suffices to test the consistency of the non-temporal theories $\Gamma_k(\Phi, \sigma)$ for each sequence σ of size $k \leq n$. In such cases, there is no relation between the size of the temporal conditions and the maximal size of the sequences to be tested. A situation where restricting the type of the formulas considered in the conditions of tcp-rules is worthwhile is when working in a context where the universe of sequences is *potentially* infinite, that is, the maximal size of the sequences evolves with time (for instance, in a temporal database context).

The following proposition relates the result stated in Theorem 4 and the result given in Proposition 2.

Proposition 3 Let Φ be a tcp-theory and $l(\Phi)$ be the length of Φ. Let us suppose that $G(\Phi)$ is acyclic and for every $\varphi \in \Phi$, $\varphi \in$ **TPref***. Then, Φ is consistent iff for every sequence σ of length $k \leq \mathrm{l}(\Phi)$, $\Gamma_k(\Phi, \sigma)$ is locally consistent. Besides, if all variables in V are binary, then consistency of Φ can be determined in time proportional to $|\Gamma|^2 \times |V| \times 2^{l(\Phi)}$.

4.4 Finding optimal sequences

In this section, given a tcp-theory Φ, we show how to determine the optimal sequences in $Seq_n(\mathcal{O})$, i.e. the maximal sequences in $Seq_n(\mathcal{O})$ with respect to $>_\Phi$ that satisfy some set of simple temporal constraints. Our approach is *incremental*, meaning that for every integer n, we show how to compute the optimal sequences in $Seq_{n+1}(\mathcal{O})$ from the set of optimal sequences in $Seq_n(\mathcal{O})$.

First, we specify the set of temporal constraints that we consider. For every integer $k > 0$, let $\mathbf{At}_k(X = a)$ be the temporal formula defined as $(\mathbf{Is}_k \wedge (X = a)) \vee \blacklozenge(\mathbf{Is}_k \wedge (X = a))$ where $\mathbf{Is}_k$ is defined by induction on k as: $\mathbf{Is}_1 = \mathbf{First}$ and $\mathbf{Is}_{i+1} = \mathbf{Prev}\ \mathbf{Is}_i$ for every integer $i > 0$. We can see that for every sequence $\sigma \in Seq(\mathcal{O})$, $\sigma \models \mathbf{At}_k(X = a)$ iff $(\sigma, k) \models (X = a)$. Intuitively, the formula $\mathbf{At}_k(X = a)$ means that in a sequence of objects, the object at state k has value a for the variable X.

In the following, we denote by **AtState** the set of formulas of the form $\mathbf{At}_k(X = a)$. Given a subset $\mathcal{C}$ of **AtState**, we say that $\mathcal{C}$ is *consistent* if there exists a sequence $\sigma \in Seq(\mathcal{O})$ such that for every $F \in \mathcal{C}$, σ satisfies F (denoted by $\sigma \models \mathcal{C}$). We can easily see that $\mathcal{C}$ is consistent iff for every pair (F, F') in $\mathcal{C}^2$ where $F = \mathbf{At}_k(X = a)$ and $F' = \mathbf{At}_{k'}(X' = a')$, if $k = k'$ and $X = X'$, then we have $a = a'$. Given a consistent subset $\mathcal{C}$ of **AtState** and an integer k, we denote by: (1) $Attr_k(\mathcal{C})$ the set of attributes $X \in V$ such that there exists a formula $\mathbf{At}_k(X = a)$ in $\mathcal{C}$. (1) $\mathcal{C}_k$ the subset of $\mathcal{C}$ defined by: $\mathcal{C}_k = \{\mathbf{At}_i(X = a) \in \mathcal{C} \mid (i \leq k)\}$. (2) $Tuple_k(\mathcal{C})$ the set of present STL formulae defined by: $Tuple_k(\mathcal{C}) = \{(X = a) \mid At_k(X = a) \in \mathcal{C}\}$.

Example 15 Let $\mathcal{C} = \{\mathbf{At}_1(G = c), \mathbf{At}_2(G = d), \mathbf{At}_2\,(D = n)\}$. It is easy to see that $\mathcal{C}$ is consistent. Moreover, $Tuple_1(\mathcal{C}) = \{(G = c)\}$ and $Tuple_2(\mathcal{C}) = \{(G = d), (D = n)\}$. Finally, we have $\mathcal{C}_1 = \{\mathbf{At}_1(G = c)\}$ and $\mathcal{C}_2 = \mathcal{C}$.

Let $\mathcal{C}$ be a consistent subset of **AtState**. Given a consistent **TPref** theory Φ, we now show how to compute for every integer n, the subset $\mathcal{S}_n(\Phi, \mathcal{C})$ of $Seq_n(\mathcal{O})$ defined by: $\mathcal{S}_n(\Phi, \mathcal{C}) = max_{>\Phi}\{\sigma \in Seq_n(\mathcal{O}) \mid \sigma \models \mathcal{C}_n\}$. The set $\mathcal{S}_n(\Phi, \mathcal{C})$ contains the optimal sequences in $Seq_n(\mathcal{O})$, i.e. the maximal sequences in $Seq_n(\mathcal{O})$ w.r.t. $>_\Phi$ that satisfy the constraints in $\mathcal{C}_n$.

Let σ_k be a sequence of length k. In the following, given the cp-theory $\Gamma = \Gamma(\Phi, \sigma_k)$ and the set of present STL formula $\mathcal{T} = Tuple_k(\mathcal{C})$, we denote by $BestObjs(\Gamma, \mathcal{T})$ the set of optimal objects in $\mathcal{O}$ that satisfy $\mathcal{T}$, i.e.

$BestObjs(\Gamma, \mathcal{T}) = max_{>\Gamma}\{o \in \mathcal{O} \mid o \models \mathcal{T}\}$. It is shown in (Wilson, 2004) how to compute this set of optimal objects.

Finally, given a tcp-theory Φ such that for every tcp-rule $\varphi \in \Phi$, $\varphi \in$ **TPref***. We can notice that for every sequence σ and σ' of length $n + 1$, if $rlo(\sigma) = rlo(\sigma')$, then $\Gamma_{n+1}(\Phi,\sigma) = \Gamma_{n+1}(\Phi,\sigma')$. Therefore, for every sequence σ of length n, we introduce the following notation: $\Gamma^*(\Phi, \sigma) = \Gamma_n(\Phi, add(\sigma, o))$ where o is any object in $\mathcal{O}$.

We now state the following theorem that shows how to compute $\mathcal{S}_{n+1}(\Phi, \mathcal{C})$ from $\mathcal{S}_n(\Phi, \mathcal{C})$.

Theorem 8 Let Φ be a consistent tcp-theory such that for every tcp-rule $\varphi \in \Phi$, $\varphi \in$ **TPref***. Let $\mathcal{C}$ be a consistent subset of $AtState$. For every sequence $\sigma = \langle o_1, \ldots, o_{n+1}\rangle \in Seq_{n+1}(\mathcal{O})$, σ is in $\mathcal{S}_{n+1}(\Phi, \mathcal{C})$ iff $rlo(\sigma) \in \mathcal{S}_n(\Phi, \mathcal{C})$ and $o_{n+1} \in BestObjs(\Gamma, \mathcal{T})$ where $\Gamma = \Gamma^*(\Phi, rlo(\sigma))$ and $\mathcal{T} = Tuple_{n+1}(\mathcal{C})$.

Proof Assume that $\sigma = \langle o_1, \ldots, o_{n+1}\rangle$ is in $\mathcal{S}_{n+1}(\Phi, \mathcal{C})$. Let $\sigma_n = rlo(\sigma)$. If $\sigma_n \notin \mathcal{S}_n(\Phi, \mathcal{C})$, it means that there exists a sequence $\sigma'_n \in \mathcal{S}_n(\Phi, \mathcal{C})$ such that $\sigma'_n \models \mathcal{C}$ and $\sigma'_n >_\Phi \sigma_n$. Since $\sigma'_n >_\Phi \sigma_n$, there exists an IFS from σ'_n to σ_n w.r.t. Φ, i.e. there exist a set of sequences $\{\tau_1, \ldots, \tau_{p+1}\}$ and a set of tcp-rules $\{\varphi_1, \ldots, \varphi_p\}$ such that $\tau_1 = \sigma'_n$, $\tau_{p+1} = \sigma_n$ and for every $k \in \{1, \ldots, p\}$, $\tau_k\ R_{\varphi k}\ \tau_{k+1}$. For every $k \in \{1, \ldots, p + 1\}$, let $\tau'_k = add(\tau_k, o_{n+1})$. Since for every tcp-rule $F_{\varphi k} = F^-_{\varphi k} \wedge F^0_{\varphi k}$ and $\tau'_k\ R_{\varphi k}\ \tau'_{k+1}$, we also have $\tau'_k\ R_{\varphi k}\ \tau'_{k+1}$. Thus, since $\tau'_{p+1} = \sigma$, there exists an IFS from τ'_1 to σ w.r.t. Φ, i.e. $\tau'_1 >_\Phi \sigma$. Moreover, we can easily see that $\tau'_1 \models \mathcal{C}$. Thus, we have $\tau'_1 >_\Phi \sigma$ and $\tau'_1 \models \mathcal{C}$ which contradicts the fact that $\sigma \in \mathcal{S}_{n+1}(\Phi, \mathcal{C})$.

On the other hand, assume that $o_{n+1} \notin BestObjs(\Gamma, \mathcal{T})$. It means that there exists an object $o'_{n+1} \in BestObjs(\Gamma, \mathcal{T})$ such that $o'_{n+1} \models \mathcal{T}$ and $o'_{n+1} >_\Gamma o_{n+1}$. Let $\sigma' = add(\sigma_n, o'_{n+1})$. We can easily show that $\sigma' >_\Phi \sigma$ and $\sigma' \models \mathcal{C}$ where $\sigma' = add(\sigma_n, o'_{n+1})$ which contradicts the hypothesis that σ is in $\mathcal{S}_{k+1}(\Phi, \mathcal{C})$. Thus, we have proved that if σ is in $\mathcal{S}_{n+1}(\Phi, \mathcal{C})$, then $rlo(\sigma) \in \mathcal{S}_n(\Phi, \mathcal{C})$ and $o_{n+1} \in BestObjs(\Gamma, \mathcal{T})$.

Conversely, assume that $\sigma_n = rlo(\sigma) \in \mathcal{S}_n(\Phi, \mathcal{C})$ and $o_{n+1} \in BestObjs(\Gamma, \mathcal{T})$. If σ is not in $\mathcal{S}_{n+1}(\Phi, \mathcal{C})$, then there exists a sequence $\sigma' = \langle o'_1, \dots, o'_{n+1}\rangle \in \mathcal{S}_{n+1}(\Phi, \mathcal{C})$ such that $\sigma' >_\Phi \sigma$ and $\sigma' \models \mathcal{C}$. Using Proposition 2, we now distinguish two cases:

- If $rlo(\sigma') >_\Phi rlo(\sigma)$, then it is easy to see that we also have $rlo(\sigma') \models \mathcal{C}_n$. Thus $rlo(\sigma') >_\Phi \sigma_n$ and $rlo(\sigma') \models \mathcal{C}_n$, which contradicts the fact that $\sigma_n \in \mathcal{S}_n(\Phi, \mathcal{C})$.
- If $rlo(\sigma') = rlo(\sigma')$, then $o'_{n+1} >_\Gamma o_{n+1}$ with $\Gamma = \Gamma_{n+1}(\Phi,\sigma_{n+1})$. Moreover, we can easily see that $o'_{n+1} \models \mathcal{T}$ since $\sigma' \models \mathcal{C}$. Thus, we have $o'_{n+1} >_\Gamma o_{n+1}$ and $o'_{n+1} \models \mathcal{T}$, which contradicts the fact that $o_{n+1} \in BestObjs(\Gamma, \mathcal{T})$.

Thus, we show that if $rlo(\sigma) \in \mathcal{S}_n(\Phi, \mathcal{C})$ and $o_{n+1} \in BestObjs(\Gamma, \mathcal{T})$, then σ is in $\mathcal{S}_{n+1}(\Phi, \mathcal{C})$, which completes the proof.

Using Theorem 8, it is easy to see that for every consistent tcp-theory and every consistent subset $\mathcal{C}$ of **AtState**, we have $\mathcal{S}_n(\Phi, \mathcal{C}) = BestSeqs(\langle\rangle,\Phi, \mathcal{C}, n)$ where $\langle\rangle$ represents the empty sequence and *BestSeqs* is the algorithm presented in Figure 5.

```
Function $BestSeqs(\sigma, \Phi, \mathcal{C}, n)$

Input:   A sequence $\sigma$ of length $k$ such that $\sigma \models \mathcal{C}_k$
         A consistent tcp-theory $\Phi$
         A consistent subset $\mathcal{C}$ of AtState
         An integer $n > k$
Output:  The set of optimal sequences $\mathcal{S}$ of size $n$ and
         prefix $\sigma$ that satisfy $\mathcal{C}_n$

1.   Let $\mathcal{S} = \emptyset$ and $k = |\sigma|$
2.   If $(k < n)$
3.       Let $\Gamma = \Gamma^*(\Phi, \sigma)$
4.       Let $\mathcal{T} = Tuple_{k+1}(\mathcal{C})$
5.       For every $o_{k+1} \in BestObjs(\Gamma, \mathcal{T})$ do
6.           Let $\sigma' = add(\sigma, o_{k+1})$
7.           $\mathcal{S} = \mathcal{S} \cup BestSeqs(\sigma', \Phi, \mathcal{C}, n)$
8.       End for
9.       Return $\mathcal{S}$
10.  Else
11.      Return $\mathcal{S}$
```

Fig. 5. Computation of Optimal Sequences

Example 16 Let $\Phi = \{\varphi_1, \dots, \varphi_6\}$ be the tcp-theory presented in our Running Example. Let $\mathcal{C} = \{At_1(G = c), At_3(D = w)\}$. We show in this example how the set $\mathcal{S}_3(\Phi, \mathcal{C})$ is computed using the algorithm presented Figure 5. Initially, we compute $\mathcal{S} = BestSeqs(\langle\rangle,\Phi, \mathcal{C}, 3)$. First, we have $\Gamma_1 = \Gamma^*(\Phi, \langle\rangle) = \{\varphi_1, \varphi_2\}$ since $F_{\varphi 1}$ and $F_{\varphi 2}$ are STL formulae in *Present*. Moreover, we have $\mathcal{T}_1 = Tuple_1(\mathcal{C}) = \{(G = c)\}$. Thus, we compute $BestObjs(\Gamma_1, \mathcal{T}_1) = \{o_1\}$ where $o_1 = (G = c, D = w)$. Then, we build the sequence $\sigma'_1 = add(\langle\rangle, o'_1) = \langle o_1\rangle$ and compute $\mathcal{S} = BestSeqs(\sigma'_1,\Phi, \mathcal{C}, 3)$.

Computing $\mathcal{S}$ = *BestSeqs*(σ'_1,Φ, $\mathcal{C}$, 3), we successively obtain $\Gamma_2 = \Gamma^*(\Phi, \langle o_1 \rangle) = \{\varphi_1, \varphi_2, \varphi_4^0\}$, $\mathcal{T}_2$ = *Tuple*$_2$($\mathcal{C}$) = 3 and *BestObjs*(Γ_2, $\mathcal{T}_2$) = {o_2} where o_2 = (G = d,D = n). Thus, we build the sequence σ'_2 = *add*($\langle o_1 \rangle$, o_2) = $\langle o_1, o_2 \rangle$ and compute $\mathcal{S}$ = *BestSeqs*(σ'_2,Φ, $\mathcal{C}$, 3).

Then, computing $\mathcal{S}$ = *BestSeqs*(σ'_2 ,Φ, $\mathcal{C}$, 3), we successively obtain $\Gamma_3 = \Gamma^*(\Phi, \langle o_1, o_2 \rangle) = \{\varphi_1, \varphi_2, \varphi_6^0\}$, $\mathcal{T}_3$ = *Tuple*$_3$($\mathcal{C}$) = {(D = w)} and *BestObjs*(Γ_3, $\mathcal{T}_3$) = {o_3} where o_3 = (G = d,D = w). Thus, we build the sequence σ'_3 = *add*($\langle o_1, o_2 \rangle$, o_3) = $\langle o_1, o_2, o_3 \rangle$ and compute $\mathcal{S}$ = *BestSeqs*(σ'_3 ,Φ, $\mathcal{C}$, 3).

Since $|\sigma'_3|$ = 3, we finally obtain $\mathcal{S}$ = *BestSeqs*($\langle\rangle$,Φ, $\mathcal{C}$, 3) = $\{\langle o_1, o_2, o_3 \rangle\}$. Note that in this example, we only obtain one optimal sequence. In general, we can obtain a set of optimal sequences since $>_\Phi$ is a *partial* order.

5. Conclusion and further research

In this chapter, we have presented several approaches for treating preferences over objects, sets of objects and sequences of objects. The main contribution is centered in Section 4 which presents a method for preference elicitation and reasoning over sequence of objects. An algorithm for finding the most preferred sequences satisfying a set of temporal constraints is introduced. A lot of work has to be done to improve our approach. (1) Concerning the algorithm for finding the best sequences, we intend to generalize our method in order to treat more general temporal constraints. (2) Concerning the expressivity power of our preference language: we note that in TPref the temporal aspect is related only to the rule conditions, that is, only to the left side of the preference rules. We are not able, for the time being, to treat preference statements such as *I prefer "this" before "that"*. (3) Concerning the consistency test: we must investigate methods to ensure consistency when the temporal conditions involve both past and future operators. (4) Concerning dominance queries: we have to investigate efficient methods to determine, given two sequences, which is the preferred one. That implies investigating efficient methods to decide, given two sequence, if there exists a IFS between them. (5) Finally, concerning a database context, the work proposed in this paper is a first step towards incorporating a formalism for *reasoning* with preferences over sequences of objects into a temporal relational query language, and so, building a bridge between the two disciplines (AI and Temporal Databases).

6. References

Bacchus, F.; Boutilier, C. & Grove, A. (1996). Rewarding behaviors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp 1160–1167, Portland, Oregon, USA, 1996. AAAI Press / The MIT Press.

Bacchus, F. & Grove, A. (1995). Graphical models for preference and utility. *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 3–10, Montreal.

Bacchus, F. & Grove, A. (1996). Utility independence in qualitative decision theory, Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning, pp. 542–552, Cambridge.

Bienvenu, M.; Fritz, C. & McIlraith, S. A. (2006). Planning with qualitative temporal preferences. *KR*, 134–144, 2006.

Boutilier, C.; Brafman, R.; Geib, C. & Poole, D. (1997). A constraint-based approach to preference elicitation and decision making, *AAAI Spring Symposium on Qualitative Decision Theory*, Stanford, 1997.

Boutilier, C.; Brafman, R.; Hoos, H. & Poole, D. (2004). Cp-nets: A tool for representing and reasoning about conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

Brafman, R.; Domshlak, C. & Shimony, S.E. (2006a). On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research*, 25:389–424, 2006.

Brafman, R.; Domshlak, C. & Shimony, S.E. (2006b). Preferences over sets. *AAAI*, 2006.

Chomicki, J. (2003). Preference formulas in relational queries. *ACM Transactions on Database Systems*, pp. 427–466, 2003.

de Amo, S. & Giacometti, A. (2007). Temporal Conditional Preferences over Sequences of Objects. *19th IEEE International Conference on Tools with Artificial Intelligence*, Patras, Greece, pp. 246-253, 2007.

des Jardins, M. & Wagstaff, K. (2005). Dd-pref: A language for expressing preferences over sets. *AAAI*, pp. 620–626, 2005.

Domshlak, C. & Brafman, R. (2002). CP-nets - reasoning and consistency testing, *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning*, pp. 121–132, Toulouse, France, 2002.

Doyle, J.; Shoham, Y. & Wellman, M. (1991). A logic of relative desire (preliminary report), *Proceedings of the Sixth International Symposium on Methodologies for Intelligent Systems (ISMIS 91)*, Lecture Notes in Computer Science, pp. 16–31. Springer-Verlag, 1991.

Endres, M. & Kießling, W. (2006). Transformation of tcp-net queries into preference database queries. *Proceedings of the ECAI 2006 Multidisciplinary Workshop on Advances in Preference Handling Riva del Garda*, Italy, August 2006, pp. 23–30.

Gabbay, D. M. (1989). The declarative past and imperative future: Executable temporal logic for interactive systems. *Lecture Notes in Computer Science*, Volume 398, pp 67–89.

Springer-Verlag, 1989. Hodksinson, I. & Reynolds, M. (2005). Separation – past, present, and future. *We Will Show Them! Essays in Honour of Dov Gabbay*, Volume 2, 2005.

Khatib, L.; Morris, P.; Morris, R.A. & Rossi, F. (2001). Temporal Constraint Reasoning With Preferences. *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pp. 322–327, 2001.

Kießling, W. (2002). Foundations of preferences in database systems. *Proceedings of 28th International Conference on Very Large Data Bases*, Hong Kong, China, pp. 311–322, 2002.

Kießling, W. & Köstler, G. (2002). Preference SQL - design, implementation, experiences. *Proceedings of 28th International Conference on Very Large Data Bases*, Hong Kong, China, pp. 990–1001, 2002.

Kumar, T.K.S. (2007). Fast (Incremental) Algorithms for Useful Classes of Simple Temporal Problems with Preferences. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, 1988.

Prestwich, S. D.; Rossi, F.; Venable, K. B. & Walsh, T. (2005). Constraint-Based Preferential Optimization. *AAAI 2005*, pp. 461–466, 2005.

Prior, A. N. (1997). *Past, Present and Future*, Oxford: Clarendon Press, 1967.

Sistla, A. P. & Clarke, E.M. (1985). The complexity of propositional linear temporal logic. *Journal of the ACM*, 32(3), pp 733–749, 1985.

Son, T.C. & Pontelli, E. (2006). Planning with preferences using logic programming. *TPLP*,6(5):559–607, 2006.

Wilson, N. (2004). Extending cp-nets with stronger conditional preference statements. *AAAI*, pp. 735–741, 2004.

Competency-based Learning Object Sequencing using Particle Swarms

Luis de Marcos, Carmen Pagés, José Javier Martínez and José Antonio Gutiérrez
University of Alcalá.
Spain

1. Introduction

Brusilovsky (1999) envisaged Web-based adaptive courses and systems as being able to achieve some important features including the ability to substitute teachers and other students support, and the ability to adapt to (and so be used in) different environments by different users (learners). These systems may use a wide variety of techniques and methods. Among them, curriculum sequencing technology is "to provide the student with the most suitable individually planned sequence of knowledge units to learn and sequence of learning tasks […] to work with". These methods derive from the adaptive hypermedia field (Brusilovsky, 1996) and rely on complex conceptual models, usually driven by sequencing rules (De Bra et al., 1999; Karampiperis, 2006). E-learning traditional approaches and paradigms, that promote reusability and interoperability, are generally ignored, thus resulting in (adaptive) proprietary systems (such as AHA! (De Bra et al., 2003)) and non-portable courseware.
On the other side, traditional approaches promote standards usage to ensure interoperability but they lack of flexibility which is in increasing demand. "In offering flexible [e-learning] programmes, providers essentially rule out the possibility of having instructional designers set fixed paths through the curriculum" (van den Berg et al., 2005). But offering personalized paths to each learner will impose prohibitive costs to these providers, because sequencing process is usually performed by instructors. So, "it is critical to automate the instructor's role in online training, in order to reduce the cost of high quality learning" (Barr, 2006) and, among these roles, sequencing seems to be a priority.
In this chapter an innovative sequencing technique that automates teacher´s role is proposed. E-Learning standards and the learning object paradigm are encouraged in order to promote and ensure interoperability. Learning units' sequences are defined in terms of competencies in such a way that sequencing problem can be modelled like a classical Constraint Satisfaction Problem (CSP) and Artificial Intelligent (AI) approaches could be used to solve it. Particle Swarm Optimization (PSO) is an AI technique and it has proven with a good performance for solving a wide variety of problems. So, PSO is used to find a suitable sequence within the solution space respecting the constraints. In section 2, the conceptual model for competency-based learning object sequencing is presented. Section 3 describes the PSO approach for solving the problem. Section 4 presents the results obtained

from the intelligent algorithm implementation and testing in a real world situation (course sequencing in an online Master in Engineering program). And finally, in Section 5 conclusions are summarized and future research lines are presented.

2. Competency-based sequencing

Within e-learning, the learning object paradigm drives almost all initiatives. This paradigm encourages the creation of small reusable learning units called Learning Objects (LOs). These LOs are then assembled and/or aggregated in order to create greater units of instruction (lessons, courses, etc) (Wiley, 2000).

LOs must be arranged in a suitable sequence prior to its delivery to learners. Currently, sequencing is performed by instructors who do not create a personalized sequence for each learner, but instead they create generic courses, which are targeted to generic learner profiles. Then, these sequences are coded using a standard specification to ensure interoperability. The most commonly used specification is SCORM (ADL, 2004). Courseware that conforms to SCORM´s Content Aggregation Model is virtually portable among a wide variety of Learning Management Systems (LMSs). Though, SCORM usage hinders the automatic LO sequencing due to its system-centered view. Other metadata-driven approaches offer better possibilities i.e. just LO metadata will enable automatic sequencing process to be performed, and the appropriate combination of metadata and competencies will allow personalized and automatic content sequencing. This section describes how to overcome these problems by defining a conceptual data model for learning object sequencing through competencies.

2.1 Competency definition

As for many other terms, there are a wide variety of definitions that try to catch the essence of the word competency in the e-learning environment. The confusion has even been increased by the work developed, often independently, in the three main fields that are nowadays primarily concerned with competencies, namely, pedagogy, human resources management and computer science. Anyway, we consider competencies as "multidimensional, comprised of knowledge, skills and psychological factors that are brought together in complex behavioural responses to environmental cues" (Wilkinson, 2001). This definition emphasizes that competencies are not only knowledge but a set of factors and that competencies are employed (bring together) in real or simulated contexts (or environments). Conceptual models for competency definitions also use to consider this multidimensionality. As an example, RDCEO specification (IMS, 2002a) describes a competency as four-dimensional element (fig. 1).

The competency 'Definition' is the record that contains general information about the competency. Each competency can be exhibited in one or more different 'Contexts'. And a set of factual data must be used to 'Evidence' that an individual has or has not acquired a particular competency. Finally 'Dimensions' are used to relate each context with its particular evidence and to store relation information such as the proficiency level.

Some e-learning trends (RDCEO have just been mentioned) are trying to formalize competency definitions. It is worth quoting the following specifications: (1) IMS "Reusable Definition of Competency or Educational Objective" (RDCEO) specification (IMS, 2002b), (2) IEEE Learning Technology Standards Committee (LTSC) "Draft Standard for Learning

Technology - Standard for Reusable Competency Definitions " specification (currently an approved draft) (IEEE, 2008), (3) HR-XML Consortium "Competencies (Measurable Characteristics) Recommendation" (HR-XML, 2006) and (4) CEN/ISSS "A European Model for Learner Competencies" workshop agreement (CEN/ISSS, 2006).

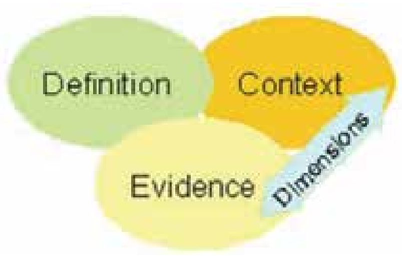


Fig. 1. RDCEO competency conceptual model (from (IMS, 2002a))

Every specification offers its own understanding of what a competency is (i.e. the definition of competency) plus a formal way to define competencies (i.e. competency definitions) so that they can be interchanged and processed by machines. A deeper analysis of these recommendations shows that, although they do not present great differences in its own definition of competency, great dissimilarities arise when the information that must conform a competency definition are confronted. In this way, it could be said that IMS and IEEE specifications are minimalist recommendations that define a small set of fields that the competency definitions should contain (in fact, only an identifier and a name are required for a conformant record). Deeper definitions of some dimensions that concern competencies (namely evidence and context) are left without specification or free to developers' interpretation. On the other hand, HR-XML specification provides competency users with a huge set of entities, fields and relations that they must fulfil in order to get conformant competency records (although many of them are optional too).

For the purpose of our study we just needed a universal way to define, identify and access to competency definitions and that is exactly what RDCEO specification offers. Moreover, RDCEO is also the oldest specification and so the most used (and the most criticized). These factors lead us to employ RDCEO records for our competency definitions. Code fragment 1 shows a sample RDCEO competency record.

```
<?xml version="1.0" encoding="utf-8"?>
<rdceo xsi:schemaLocation="http://www.imsglobal.org/xsd/imsrdceo_rootv1p0"
xmlns="http://www.imsglobal.org/xsd/imsrdceo_rootv1p0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <identifier>
     http://www.uah.es/cc/comps/CompsTaxon.xml#1IntroWeb
   </identifier>
   <title>
      <langstring xml:lang="en">
        Web, Internet and Distributed Systems Introduction
      </langstring>
   </title>
</rdceo>
```

Code 1. Sample Competency Record.

2.2 Competencies for interoperable learning object sequencing

According to RDCEO and IEEE nomenclature, a competency record is called 'Reusable Competency Definition' (or RCD). RCDs can be attached to LOs in order to define its prerequisites and its learning outcomes. We have used this approach to model LO sequences. By defining a competency (or a set of competencies) as a LO outcome, and by identifying the same competency as the prerequisite for another LO (fig. 2), a constraint between the two LOs is established so that the first LO must precede the second one in a valid sequence.

Meta-Data (MD) definitions are attached to LOs, and within those definitions references to competencies (prerequisites and learning outcomes) are included. LOM (IEEE, 2002) records have been used for specifying LO Meta-Data. LOM element 9, 'Classification', is used to include competency references as recommended in by IMS (2002a). So, LOM element 9.1, 'Purpose', is set to 'prerequisite' or 'educational objective' from among the permitted vocabulary for this element; and LOM element 9.2 'Taxon Path', including its sub-elements, is used to reference the competency. Note that more than one 'Classification' element can be included in one single LO in order to specify more than one prerequisite and/or learning outcome. In code fragment 2 it is shown a sample LO metadata record that holds two competency references, a prerequisite relation and a learning outcome relation.

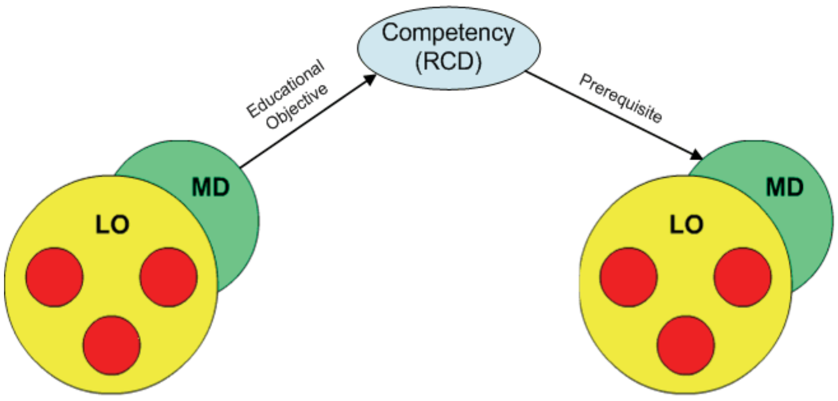


Fig. 2. LO sequencing through competencies

```
<?xml version="1.0" encoding="iso-8859-1"?>
    <lom:lom xmlns:lom="http://ltsc.ieee.org/xsd/LOM">
      <lom:general>
        <lom:title>
          <lom:string language="en">HTML</lom:string>
        </lom:title>
        <lom:language>en</lom:language>
        <lom:description>
          <lom:string language="en">HTML Course</lom:string>
        </lom:description>
      </lom:general>
      <lom:lifeCycle>
        <lom:version>
          <lom:string language="en">1.0</lom:string>
        </lom:version>
        <lom:contribute>
          <lom:date>
            <lom:dateTime>2007-01-10</lom:dateTime>
          </lom:date>
        </lom:contribute>
      </lom:lifeCycle>
      <lom:educational>
        <lom:difficulty>
          <lom:value>easy</lom:value>
        </lom:difficulty>
        <lom:typicalLearningTime>
          <lom:duration>PT50H</lom:duration>
        </lom:typicalLearningTime>
        <lom:language>en</lom:language>
      </lom:educational>
      <lom:classification>
        <lom:purpose>prerequisite</lom:purpose>
        <lom:taxonPath>
          <lom:source>
            <lom:string language="en">
              http://www.uah.es/cc/comps/CompsTaxon/
            </lom:string>
          </lom:source>
          <lom:id>1IntroWeb</lom:id>
        </lom:taxonPath>
      </lom:classification>
      <lom:classification>
        <lom:purpose>educational objective</lom:purpose>
        <lom:taxonPath>
          <lom:source>
            <lom:string language="en">
              http://www.uah.es/cc/comps/CompsTaxon/
            </lom:string>
          </lom:source>
          <lom:id>3HTML</lom:id>
        </lom:taxonPath>
      </lom:classification>
    </lom:lom>
```

Code 2. Sample LO metadata record containing competency references

Simple metadata (i.e. LOM records) is enough to model LOs' sequences in a similar way. Then, Why use competencies? Competency usage is encouraged, besides its usefulness for modelling prerequisites and learning outcomes, because competencies are also useful for modelling user current knowledge and learning initiatives' expected outcomes (future learner knowledge).We are proposing a wider framework (fig. 3) in which learner (user) modelling is done in terms of competencies, which are also used to define the expected learning outcomes from a learning program. Both sets of competencies constitute the input for a gap analysis process. This process performs a search in local and/or distributed remote repositories in order to identify the set of learning objects that fill the gap between learner current knowledge and the learning objectives. Gap analysis process returns a set of unordered LOs that must be assembled and structured in a comprehensive way, so that basic units (LOs) are presented to the learner previously to advanced lessons. These actions will be performed by the LO sequencing process depicted in figure 3.

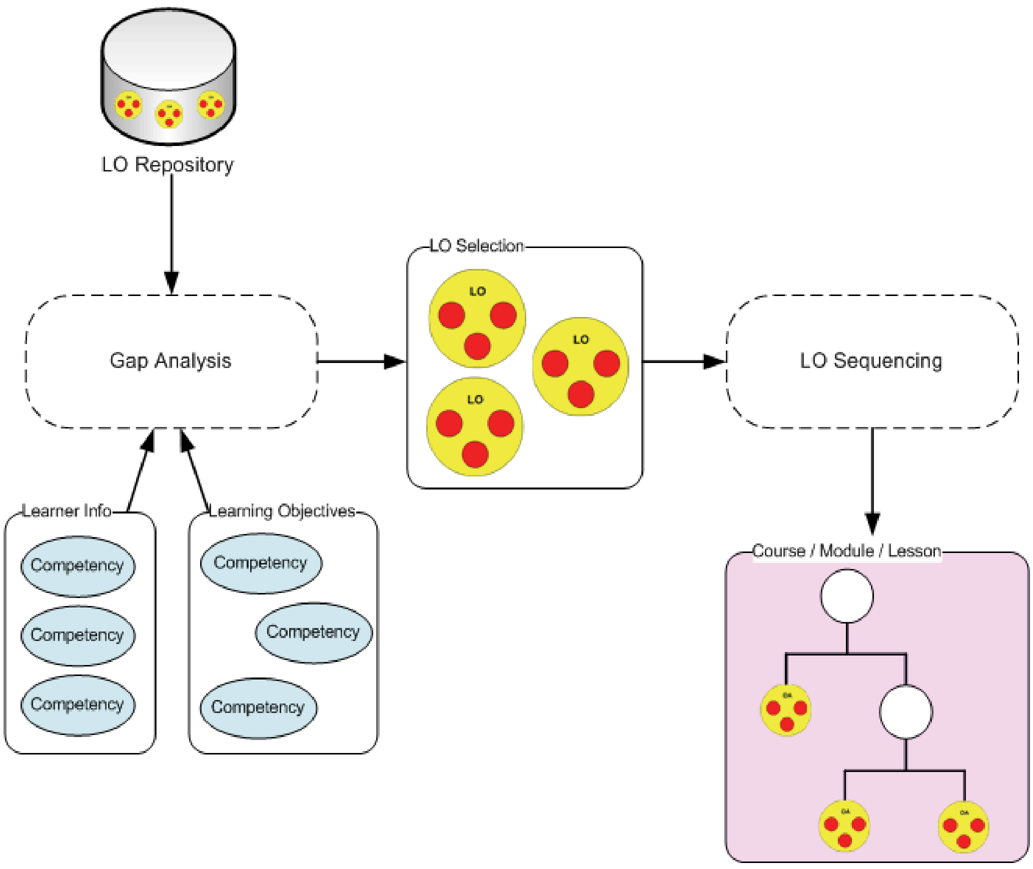


Fig. 3. Competency-driven content generation model

3. Competency-based intelligent sequencing

Given a random LOs' sequence modelled as described above (with competencies representing LOs prerequisites and learning outcomes), the question of finding a correct sequence can be envisaged as a classical artificial intelligent Constraint Satisfaction Problem (CSP). In this way, the solution space comprises all possible sequences (*n!* will be its size, total number of states, for *n* LOs), and a (feasible) solution is a sequence that satisfies all established constraints. LO permutations inside the sequence are the operations that define transitions among states. So we face a permutation problem, which is a special kind of CSP. PSO is an AI evolutionary computing technique that can be used to solve CSP problems (among other kind of problems). This section presents a mathematical characterization of the learning object sequencing problem so that a PSO implementation can be formally specified. Then this PSO implementation is presented and some improvements over the original algorithm are proposed.

3.1 Mathematical characterization

According to (Tsang, 1993) a CSP is a triple *(X,D,C)* where $X = \{x_o, x_1,\ldots,x_{n-1}\}$ is finite set of variables, *D* is a function that maps each variable to its corresponding domain *D(X)*, and

$C_{i,j} \subset D_i \times D_j$ is a set of constraints for each pair of values *(i, j)* with $0 \leq i < j < n$. To solve the CSP is to assign all variables x_i in *X* a value from its domain *D*, in such a way that all constraints are satisfied. A constraint is satisfied when $(x_i, x_j) \in C_{i,j}$ and (x_i, x_j) it is said to be a valid assignment. If$(x_i, x_j) \notin C_{i,j}$ then the assignment (x_i, x_j) violates the constraint.
If all solutions from a CSP are permutations of a given tuple then it is said that the problem is a permutation CSP or PermutCSP. A PermutCSP is defined by a quadruple *(X,D,C,P)* where *(X,D,C)* is a CSP and $P=<v_0, v_1, \ldots, v_{n-1}>$ is a tuple of $|X|=n$ values. A solution S of a PermutCSP must be a solution of *(X,D,C)* and a complete permutation of P.
The learning object sequencing problem could be modeled as a PermutCSP. For example, considering five learning objects titled 1,2,3,4 and 5, the PermutCSP which only solution is the set S = {1,2,3,4,5} (all learning objects must be ordered) can be defined as:

$$X = \{x_1, x_2, x_3, x_4, x_5\}$$

$$D(X_i) = \{1,2,3,4,5\} \ \forall x_i \in X$$

$$C = \{x_{i+1} - x_i > 0 : x_i \in X, i \in \{1,2,3,4\}\}$$

$$P = <1,2,3,4,5>$$

As it will be demonstrated later a good definition of the constraint set *C* critically affects the solving algorithm performance and even its completeness.

3.2 Particle swarm optimization

Particle Swarm Optimization (PSO) is an evolutionary computing optimization algorithm. PSO mimics the behaviour of social insects like bees. A random initialized particles' population (states) flies through the solution space sharing the information they gather. Particles use this information to dynamically adjust its velocity and cooperate towards finding a solution. Best solution found: (1) by a particle is called *pbest*, (2) within a set of neighbour particles is called *nbest*, (3) and within the whole swarm is called *gbest*. Goodness of each solution is calculated using a function called fitness function. A basic PSO procedure, adapted from (Hu et al., 2003), is showed in code fragment 3. PSOs have been used to solve a wide variety of problems (Hinchey et al., 2007).
The original PSO (Eberhart & Kennedy, 1995; Kennedy & Eberhart, 1995) is intended to work on continuous spaces, and velocity is computed for each dimension $x_i \in \overline{X}$. Particles' initial position and initial velocity are randomly assigned when the population (swarm) is initialized. A discrete binary version of the PSO was presented by Kennedy and Eberhart (1997). This version uses the concept of velocity as a probability of changing a bit state from zero to one or vice versa. A version that deals with permutation problems was introduced by Hu et al., (2003). In this latter version, velocity is computed for each element in the sequence, and this velocity is also used as a probability of changing the element, but in this case, the element is swapped establishing its value to the value in the same position in *nbest*. Velocity is updated using the same formula for each variable in the permutation set $(x_i \in X)$, but it is also normalized to the range 0 to 1 by dividing each x_i by the maximum range of the particle (i.e. maximum value of all $x_i \in X$). The mutation concept is also introduced in this permutation PSO version; after updating each particle´s velocity, if the current particle is equal to *nbest* then two randomly selected positions from the particle

sequence are swapped. Hu et al., (2003) have also demonstrated that permutation PSO outperforms genetic algorithms for the N-Queens problem. So we decided to try PSO, before any other technique, for the LO sequencing problem.

```
initialize the population
do {
 for each particle {
  calculate fitness value
  if (new fitness > pBest)
   set pbest = current value
 }
 nbest = particle with the best fitness value of all the topological neighbors
 for each particle {
  Calculate new velocity as
   $\overline{V}_{new}$=w x $\overline{V}_{old}$ + c1 x rand () x ($\overline{P}_{best}$ - $\overline{X}$) + c2 x rand() x ($\overline{P}_{nbest}$- $\overline{X}$)
  Update particle position
   $\overline{X}_{new}$= $\overline{X}_{old}$ + $\overline{V}_{new}$
 }
} until termination criterion is met
```

Code 3. PSO Procedure Pseudo-code

rand() is a function that returns a random number between 0 and 1. Each instance of *rand()* in the algorithm represents a new call to the function, i.e. a new random number is computed and returned.

Each particle shares its information with a, usually fixed, number of neighbor particles to determine *nbest* value. Determining the number of neighbor particles (the neighbor size) and how neighborhood is implemented has been a subject of deep research in an area that has been called sociometry. Topologies define structures that determine neighborhood relations, and several of them (ring, four cluster, pyramid, square and all topologies) have been studied. It has been proved that fully informed approaches outperform all other methods (Mendes et al., 2004). The fully informed approach prompts using an 'all' topology and a neighborhood size equal to the total number of particles in the swarm (i.e. every particle is connected with all other particles when *nbest* values are calculated, hence *gbest* is always equal to *nbest*).

3.3 PSO for learning object sequencing

Discrete full-informed version of the PSO was implemented in order to test its performance for solving the LO sequencing problem. Code fragment 4 shows the basic procedure for LO sequencing pseudo code. Several other issues concerning design and implementation have to be decided. In the rest of this section each of these issues is discussed and the selection criteria are explained.

Fitness Function. It is critical to choose a function that accurately represents the goodness of a solution (Robinson & Rahmat-Samii, 2004). In PSO, like in other evolutionary techniques algorithms and meta-heuristics search procedures, there is usually no objective function to be maximized. A common used fitness function when dealing with CSP problems is a standard penalty function (Schoofs & Naudts, 2000):

$$f(X) = \sum_{0 \le i < j < n} V_{i,j}(x_i, x_j) \tag{1}$$

where Vi,j : Di x Dj →{0,1} is the violation function

$$V_{i,j}(\mathrm{x_i}, \mathrm{x_j}) \begin{cases} 0 \text{ if } (\mathrm{x_i}, \mathrm{x_j}) \in C_{i,j} \\ 1 \ otherwise \end{cases} \tag{2}$$

```
initialize the population
do {
  for each particle {
    calculate fitness value
    if (new fitness < gBest)
      set gbest = currentValue
    if (new fitness < pBest)
      set pbest = currentValue
    Calculate new velocity as
      $\overline{V}_{new}$ = w x $\overline{V}_{old}$ + c1 x rand() x ($\overline{P}_{pbest}$ - $\overline{X}$) + c2 x rand() x ($\overline{P}_{gbest}$ - $\overline{X}$)
    Normalize Velocity as
      $\overline{V}_{norm}$ = $\overline{V}_{new}$ /max ($\overline{V}_{new}$)
    Update particle value
      for each v[i] in $\overline{V}_{norm}$ {
        if(rand() < v[i])
          swap currentValue[i] for indexOf(currentValue, gBest[i])
      }
    Check Mutation
      if (currentValue = gBest) swap two random positions from currentValue
  }
} until termination criterion is met
```

Code 4. PSO Procedure for LO Sequencing

The standard penalty function returns the number of constraints violated, so PSO objective is to minimize that function (sentence if (new fitness > *pBest*) was changed to if (new fitness < *pBest*)). When a particle returns a fitness value of 0, a sequence that satisfies all constraints has been found and the algorithm processing is finished.

This fitness function works well if the constraint set *C* for the PermutCSP has been accurately defined. In the example presented in section 3.1 that represents a 5 LO sequence with only one feasible solution, the restriction set was defined as C={$x_{i+1}-x_i > 0$: $x_i \in$ X, i∈{1,2,3,4}}. A more accurate definition will be C= {x_i-x_j>0: $x_i \in$ X, $x_j \in \{x_1,\ldots,x_i\}$}. If we consider the sequence {2,3,4,5,1} the standard penalty function will return 1 if the first definition of *C* is used, while the returned value will be 4 if the second definition is used. The second definition is more accurate because it returns a better representation of the number of swaps required to turn the permutation into the valid solution. Moreover, the first definition of *C* has additional disadvantages because some really different sequences (in terms of its distance to the solution) return the same fitness value. For example sequences

{2,3,4,5,1}, {1,3,4,5,2}, {1,2,4,5,3} and {1,2,3,5,4} will return a fitness value of 1. Fortunately, the accurate constraint definition problem could be solved programmatically. A function that recursively processes all restrictions and calculates the most precise set of restrictions violated by a given sequence was developed and called over the input PSO sequence. This process was called the 'real' constraint calculator. The user (instructor, content provider,…) will usually define the minimum necessary number of constraints and the system will compute real constraints in order to ensure algorithm convergence, so user obligations are lightened simultaneously.

PSO Parameters. One important PSO advantage is that it uses a relatively small number of parameters compared with other techniques like genetic algorithms. However, much literature on PSO parameter selection has been written. Among it, Hu et. al. (2003) established the set of parameters in such a way that PSO works properly for solving permutation problems. So we decided to follow their recommendations, and parameters were set as follows: Learning rates (*c1*, *c2*) are set to 1.49445 and the inertial weight (*w*) is computed according to the equation (3).

$$w = 0.5 + (\text{rand}()/2) \tag{3}$$

where *rand()* represents a call to a function that returns a random number between 0 and 1. Population size was set to 20 particles. As the fully informed was used, it was not necessary to make any consideration concerning the neighborhood size.

Initialization. The algorithm receives an initial sequence *I* as an input. This input is used to initialize the first particle. All other particles are initialized randomly by permuting *I*. Initial velocity for each particle is also randomly initialized as follows: Each $v_i \in V$ is randomly assigned a value from the range $\{0,|I|\}$, where $|I|$ is the total number of learning objects in the sequence.

Termination criteria. Agent processing stops when a fitness evaluation of a particle returns 0 or when a fixed maximum number of iterations is reached. So the number of iterations was also defined as an input parameter. It was used as a measurement of the number of calls to the fitness function that were allowed to find a solution. It should be noted that some problems may not have a solution, so the number of iterations setting can avoid infinite computing.

Proposed improvements. During the initial agent development we found that in some situations the algorithm got stuck in a local minimum, and it was not able to find a feasible solution. For that reason, two enhancements were envisaged in order to improve algorithm performance for LO sequencing. First improvement was to decide randomly whether the permutation of a particle's position was performed from *gbest* or from *pbest* (p=0.5). In the original version all permutations were done regarding *gbest*. The second improvement was consisted in changing *pbest* and *gbest* values when an equal or best fitness value was found by a particle. In other words all particle's comparisons concerning *pbest* and *gbest* against the actual state were set to less or equal (<=) because the fitness function is to be minimized. The original algorithm determines that *pbest* and *gbest* only change if a better state is found (comparisons strictly <). Code fragment 5 presents the final sequencing algorithm pseudo code that includes these improvements. Changes respecting the basic procedure are showed underscored.

These changes resemble to be quite logical ways for increasing particles' mobility and for avoiding quick convergence to local minimums. And they were tested later in the results phase.

```
initialize the population
do {
 for each particle {
  calculate fitness value
  if (new fitness <= gBest)
   set gbest = currentValue
  if (new fitness <= pBest)
   set pbest = currentValue
  Calculate new velocity as
  $\overline{V}_{new} = w \times \overline{V}_{old} + c1 \times rand() \times (\overline{P}_{pbest} - \overline{X}) + c2 \times ran() \times (\overline{P}_{gbest} - \overline{X})$
  Normalize Velocity as
  $\overline{V}_{norm} = \overline{V}_{new} / _{max}(\overline{V}_{new})$
  Update particle value
   for each v[i] in $\overline{V}_{norm}$ {
    if(rand() < v[i])
     if(rand() < 0.5)
      swap currentValue[i] for currentValue[indexOf(currentValue, pBest[i]])
     else
      swap currentValue[i] for currentValue[indexOf(currentValue, gBest[i]])
   }
  Check Mutation
   if (currentValue = gBest) swap two random positions from currentValue
 }
} until termination criterion is met
```

Code 5. Improvements on PSO Procedure

4. Experimental results and discussion

The PSO algorithm for LOs sequencing described above was designed and implemented using the object oriented paradigm. We wanted to test its performance in a real scenario so a problem concerning course sequencing for a Master in Engineering (M.Eng.) program in our institution, the Computer Science School from the University of Alcalá in Madrid (Spain), was chosen for testing. The (web engineering) M.Eng, program comprises 23 courses (subjects) grouped in:

- Basic courses (7) that must be taken before any other (kind of course). There may be restrictions between two basic courses, for example 'HTML' course must precede Javascript course.
- 'Itinerary' courses (5) that must be taken in a fixed ordered sequence.
- Compulsory courses (5). There may be restrictions between two compulsory courses.
- Elective courses (6). Additional constraints with respect to any other course may be set.

All courses have an expected learning time that ranges from 30 to 50 hours. They are delivered online using a LMS, namely EDVI LMS (Barchino et al., 2005), and every course has its metadata record. Competency records were created to specify LOs' restrictions, and LOM metadata records were updated to reflect prerequisite and learning outcome competencies as detailed in section 2. A feasible sequence must have 23 LOs satisfying all constraints. The graph showing all LOs and constraints is very complex, and so it is to calculate the exact number of feasible solutions. Some estimations have been used, we have estimated that the relation among feasible solutions and total solutions order is $8{,}9\mathrm{x}10^{12}$. This number reflects the number of states (non-feasible solutions) for each feasible solution.

Once the problem was established, PSO agent parameters were set to test four different configurations that reflect all possibilities concerning proposed improvements introduced in Section 3. These configurations are:

- Configuration 1. Permutation of the particle position is randomly selected from *gbest* or from *pbest*. Comparison for changing particle *pbest* and *gbest* values is set to less or equal (<=).
- Configuration 2. Permutations from *gbest/pbest*. Comparison set to strictly less (<).
- Configuration 3. All permutations are performed from *gbest*. Comparison set to less or equal (<=).
- Configuration 4. Permutations from *gbest*. Comparison set to strictly less (<).

Figure 4 shows the results. Each configuration was run 1000 times allowing 20, 30, 40, 50, 75, 100, 150, 200, 300 and 500 iterations, and the succeed ratio was observed. From the results, it can be seen that all configurations converge to a feasible solution, but configuration 4 (original settings) outperform all others. Figure 4 also shows that original settings need less fitness evaluations. This argument is supported by table 1 results, where it is showed the mean number of evaluation function calls required for each configuration to find a solution (1000 runs) if the number of iterations parameter is set to a number high enough (i.e. a number of iterations that ensures a success ratio of 1 for each configuration).

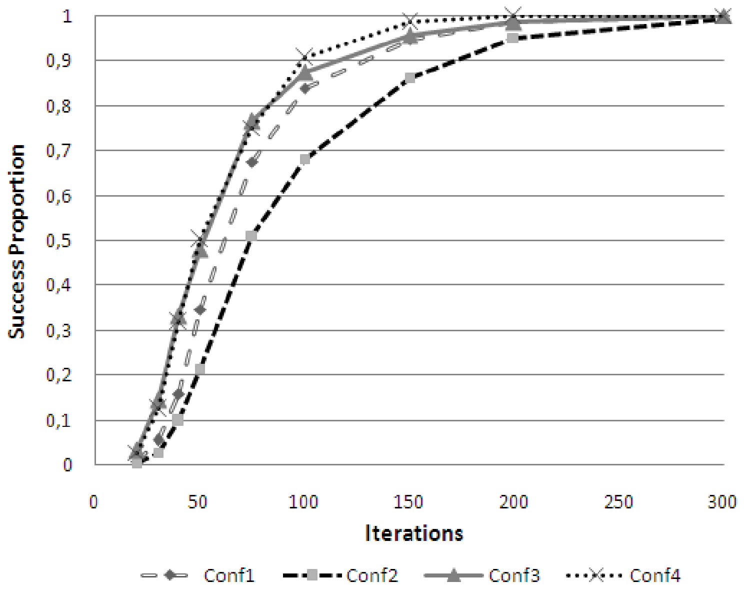


Fig. 4. PSO Configurations Comparison

An example of the PSO sequencing agent execution for the test case is shown in figure 5. The input is a random sequence of learning objects and the output is a valid sequence (i.e. a sequence that satisfies all restrictions). In the output sequence (1) all basic courses are placed in the initial positions of the sequence, (2) itinerary courses are properly ordered, and (3) compulsory, itinerary and elective courses are intercalated respecting all constraints. Output is also complemented by the number of fitness function calls required to find the solution.

The tested scenario may seem to have many feasible solutions that would make doubtful PSO performance in not-so-kind scenarios, so PSO agent was tested in 'more' difficult situations. Test sequences containing 5, 10, 20, 30, 40, 50, 60, 75 and 100 learning objects with only one feasible solution in the solution space were designed. Configuration 4 was used because it showed the best performance for the above test case and unlimited iterations were allowed to find the solution. Fitness evaluation means were observed for 100 runs (fig. 6).

Although fitness evaluations does not increase linearly to the number of learning objects, it should be noted that learning objects increment entails an exponential explosion of solution space size (remember that solution space size for *n* learning objects will be *n!*). For example, the solution space with 100 learning objects will be 10^{48} times bigger than the solution space with 75 learning objects, but the number of fitness evaluations required for finding a solution is only twice bigger. In other words, X-axis could also be interpreted as the solution space size expressed in a logarithmic scale. Therefore, the intelligent agent also handles reasonably the combinatorial explosion inherent to many AI problems.

	Fitness Evaluations
Configuration 1	1412
Configuration 2	1817
Configuration 3	1237
Configuration 4	1158

Table 1. Number of Fitness Evaluations

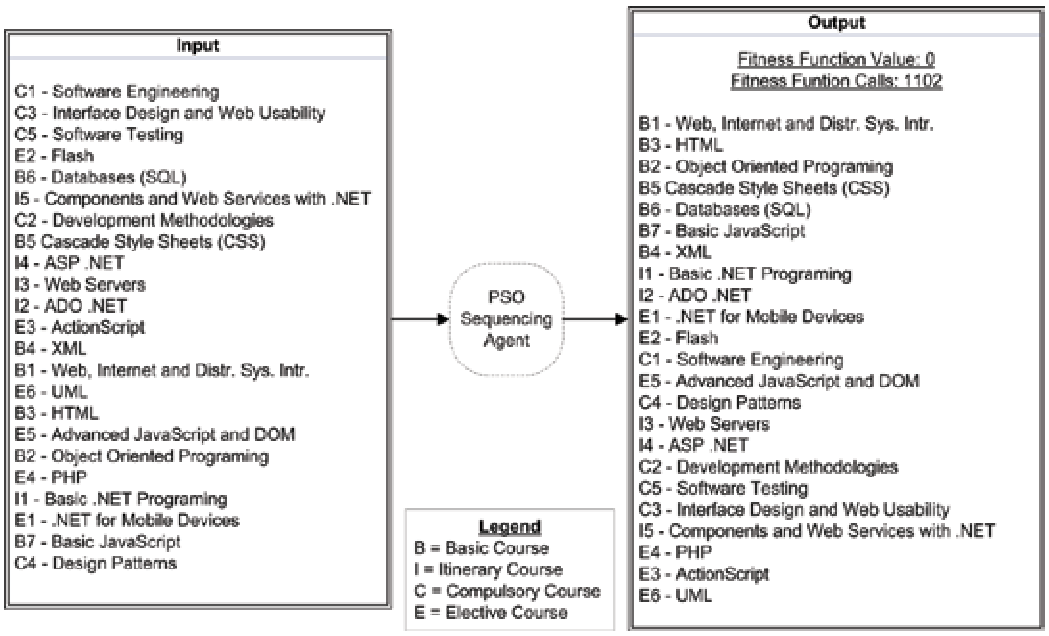

Input
C1 - Software Engineering
C3 - Interface Design and Web Usability
C5 - Software Testing
E2 - Flash
B6 - Databases (SQL)
I5 - Components and Web Services with .NET
C2 - Development Methodologies
B5 Cascade Style Sheets (CSS)
I4 - ASP .NET
I3 - Web Servers
I2 - ADO .NET
E3 - ActionScript
B4 - XML
B1 - Web, Internet and Distr. Sys. Intr.
E6 - UML
B3 - HTML
E5 - Advanced JavaScript and DOM
B2 - Object Oriented Programing
E4 - PHP
I1 - Basic .NET Programing
E1 - .NET for Mobile Devices
B7 - Basic JavaScript
C4 - Design Patterns
PSO Sequencing Agent
Legend
B = Basic Course
I = Itinerary Course
C = Compulsory Course
E = Elective Course
Output
Fitness Function Value: 0
Fitness Funtion Calls: 1102
B1 - Web, Internet and Distr. Sys. Intr.
B3 - HTML
B2 - Object Oriented Programing
B5 Cascade Style Sheets (CSS)
B6 - Databases (SQL)
B7 - Basic JavaScript
B4 - XML
I1 - Basic .NET Programing
I2 - ADO .NET
E1 - .NET for Mobile Devices
E2 - Flash
C1 - Software Engineering
E5 - Advanced JavaScript and DOM
C4 - Design Patterns
I3 - Web Servers
I4 - ASP .NET
C2 - Development Methodologies
C5 - Software Testing
C3 - Interface Design and Web Usability
I5 - Components and Web Services with .NET
E4 - PHP
E3 - ActionScript
E6 - UML

Fig. 5. PSO Agent Execution Example

5. Conclusions

Automated LO sequencing is a recurring problem in the e-learning field that could be undertaken employing models that ensure interoperability and artificial intelligent techniques. The purpose of the study was to design, develop and test a PSO agent that performs automatic LO sequencing through competencies. A model that employs competencies as a mean for defining constraints between learning object has been presented, so that a sequence of LOs is defined by relations among LOs and competencies. New sequences can be derived if permutation operations are allowed between LOs in the sequence. Hence the sequencing problem is turn into a permutation problem, and the aim is to find a sequence that satisfies all restrictions expressed in the original model. The PSO for permutation problem has been extended to LO sequencing problem. Testing two envisaged improvements was also performed. Results show that: (1) PSO succeeds in solving the problem, and (2) the original configuration is the best one.

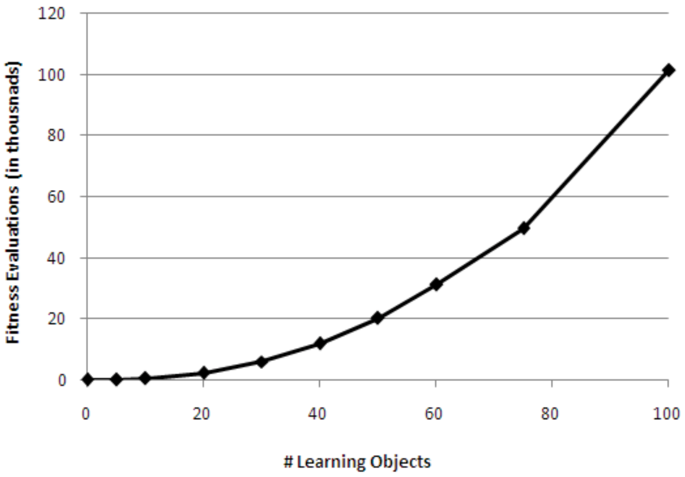


Fig. 6. Number of fitness evaluations required for different number of LOs

Further implications arise from the model proposal and from the study conclusions: (1) E-learning standards are promoted. XML records and bindings are used, so elements will be easily interchanged and processed by compliant systems. (2) Instructor's role is automated reducing costs. Sequencing process works even in complex scenarios where humans face difficulties. Instructors could spend saved time in performing other activities within the learning action. And (3), the model can be extended to an automated intelligent system for building personalized e-learning experiences. But this third implication is linked to future work. This model has been envisaged and it was depicted in figure 3 (Section 2.2). Sequencing process can be complemented with gap analysis process and competency learner modelling techniques to build personalized courses. These courses could also be SCORM (ADL, 2004) compliant, so they could be imported to current LMSs.

6. Acknowledgments

This research is co-funded by: (1) the University of Alcalá FPI research staff education program, (2) the Spanish Ministry of Industry, Tourism and Commerce PROFIT program (grants FIT-350200-2007-6 and FIT-350101-2007-9) and Plan Avanza program (grant PAV-070000-2007-103), (3) the Spanish Ministry of Education and Science PROFIT program (grant CIT-410000-2007-5), (4) Castilla-La Mancha autonomous community under the educational innovation cooperation program (grant EM2007-004) by (5) research groups' support program from the University of Alcalá and CAM Madrid Region (grant CCG06-UAH/TIC-0732).

7. References

ADL (2004) Shareable Content Object Reference Model (SCORM). The SCORM 2004 Overview. Advanced Distributed Learning (ADL) Initiative.

Barchino, R.; Gutiérrez, J. M. & Otón, S. (2005) *An Example of Learning Management System*. In Isaías, P., Baptista, M. & Palma, A. (Eds.) *IADIS Virtual Multi Conference on Computer Science and Information Systems (MCCSIS 2005).* Virtual, IADIS Press.

Barr, A. (2006) Revisiting the -ilities: Adjusting the Distributed Learning Marketplace, Again? *Learning Technology Newsletter,* 8**,** 3-4.

Brusilovsky, P. (1996) Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction,* 6**,** 87-129.

Brusilovsky, P. (1999) Adaptive and Intelligent Technologies for Web-based Education. *Künstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching,* 4**,** 19-25.

CEN/ISSS (2006) European Model for Learner Competencies. Comité Européen de Normalisation / Information Society Standardization System (CEN/ISSS).

De Bra, P.; Aerts, A.; Berden, B.; Lange, B. D.; Rousseau, B.; Santic, T.; Smits, D. & Stash, N. (2003) AHA! The adaptive hypermedia architecture. *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia.* Nottingham, UK, ACM Press.

De Bra, P., Houben, G.-J. & Wu, H. (1999) AHAM: a Dexter-based reference model for adaptive hypermedia. *Proceedings of the tenth ACM Conference on Hypertext and hypermedia.* Darmstadt, Germany, ACM Press.

Eberhart, R. & Kennedy, J. (1995) A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science. MHS '95.* Nagoya, Japan.

Hinchey, M. G., Sterritt, R. & Rouff, C. (2007) Swarms and Swarm Intelligence. *Computer,* 40**,** 111-113.

HR-XML (2006) Competencies (Measurable Characteristics) Recommendation. HR-XML Consortium.

Hu, X., Eberhart, R. C. & Shi, Y. (2003) Swarm intelligence for permutation optimization: a case study of n-queens problem. *Proceedings of the 2003 IEEE Swarm Intelligence Symposium.* Indianapolis, USA, IEEE Press.

IEEE (2002) Learning Technology Standards Committee (LTSC). Learning Object Metadata (LOM). 1484.12.1. IEEE.

IEEE (2008) Learning Technology Standards Committee (LTSC). Standard for LearningTechnology - Data Model for Reusable Competency Definitions. IEEE.

IMS (2002a) Reusable Definition of Competency or Educational Objective - Best Practice and Implementation Guide. IMS Global Learning Consortium.

IMS (2002b) Reusable Definition of Competency or Educational Objective - Information Model. IMS Global Learning Consortium.

Karampiperis, P. (2006) Automatic Learning Object Selection and Sequencing in Web-Based Intelligent Learning Systems. IN ZONGMIN, M. (Ed.) *Web-Based Intelligent E-Learning Systems: Technologies and Applications.* London. UK., Idea Group.

Kennedy, J. & Eberhart, R. (1995) Particle swarm optimization. *Proceedings., IEEE International Conference on Neural Networks.* Perth, WA, Australia.

Kennedy, J. & Eberhart, R. C. (1997) A discrete binary version of the particle swarm algorithm. *1997 IEEE International Conference on Systems, Man, and Cybernetics. 'Computational Cybernetics and Simulation'.*

Mendes, R., Kennedy, J. & Neves, J. (2004) The fully informed particle swarm: simpler, maybe better. *Evolutionary Computation, IEEE Transactions on,* 8, 204-210.

Robinson, J. & Rahmat-Samii, Y. (2004) Particle swarm optimization in electromagnetics. *Antennas and Propagation, IEEE Transactions on,* 52, 397-407.

Schoofs, L. & Naudts, B. (2000) Ant colonies are good at solving constraint satisfaction problems. *Proceedings of the 2000 Congress on Evolutionary Computation.* La Jolla, CA.

Tsang, E. (1993) *Foundations of Constraint Satisfaction,* Academic Press.

Van Den Berg, B., Van Es, R., Tattersall, C., Janssen, J., Manderveld, J., Brouns, F., Kurvers, H. & Koper, R. (2005) Swarm-based sequencing recommendations in e-learning. *Proceedings 5th International Conference on Intelligent Systems Design and Applications, 2005. ISDA '05.* Wroclaw, Poland.

Wiley, D. A. (2000) Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. IN WILEY, D. A. (Ed.) *The Instructional Use of Learning Objects.*

Wilkinson, J. (2001) A matter of life or death: re-engineering competency-based education through the use of a multimedia CD-ROM. *IEEE International Conference on Advanced Learning Technologies, 2001. Proceedings.*

Image Thresholding of Historical Documents Based on Genetic Algorithms

Carmelo Bastos Filho, Carlos Alexandre Mello, Júlio Andrade,
Marília Lima, Wellington dos Santos, Adriano Oliveira and Davi Falcão
Department of Computing and Systems, University of Pernambuco
Brazil

1. Introduction

Digital Libraries have been developed nowadays as a way to dispose digital information through the Internet. This is particularly very useful when the information comes from historical documents. This research takes place in the PROHIST Project [Mello et al., 2008] which aims the creation of a digital library with methods to preserve and broadcast images of historical documents. In general, the access to original documents has to be done carefully as, because of its age, the paper is more susceptible to the wear and tear over time. In order to make the documents more easily accessible, digitization comes as the most efficient solution. In a digital media, as digital images, the documents can be visualized and copied. This also helps the preservation of the documents as they are digitized in high resolution and in true color format. It is common to use JPEG file format (Sayood, 1996) to store these images ensuring a good space storage/quality ratio. However, even in this format, to access an archive of thousands of high quality true color images is not an easy task even with the extended use of broad band Internet.
The storage space of the images can be reduced with its conversion to black-and-white images. In this bi-level format and stored using GIF file format, the size of the file can be five times lower than the original true color JPEG image. Binarization or thresholding (Parker, 1997) is the process that converts an image into black-and-white: a threshold value is defined and the colors above that value are converted into white, while the colors below it are converted into black. This is a very simple process in digital image processing when one has a document with black ink written on a white paper. Historical documents, however, have several types of noises. The degradation yellows the sheet of paper and creates some noise that is perceptible to the digitizing process. Even more, in some cases, the ink has faded. This is particularly important when the document is written on both sides of the paper. In some cases, the ink of one side interferes in the other creating an effect called "ink bleeding". Because of these problems, it is very difficult to find the best threshold value that separates the colors that belong to the paper from the colors that belong to the ink. An example of such a document is presented in Figure 1-left.
In this paper, we present a new thresholding algorithm for color quantization based on genetic algorithms and image fidelity metrics. These metrics are used to define the convergence point of the genetic algorithm. The quantized image is then binarized based on

the number of classes defined in the quantization phase. The algorithm is adjusted to work on images of historical documents. Figure1-centre presents the final bi-level image of Figure 1-left. This is the final kind of image that we are looking for in our method. Fig. 1-right presents the results of an incorrect choice of a threshold value. The comprehension of the foreground text is severely damaged.

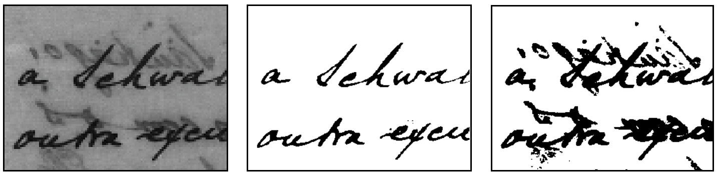


Fig. 1. (left) Zooming into part of an historical document written on both sides of the paper, presenting the ink bleeding effect, (centre) the ideal bi-level image and (right) an incorrect threshold value can create a highly noisy image.

In the next Section, we briefly review some important aspects of color quantization, genetic algorithms and image fidelity assessment. Section 3 describes our method while Section 4 presents and analyzes the results. Section 5 concludes the Chapter.

2. Fundamentals

2.1 Color quantization

Color quantization (Parker, 1997) is the process of selecting a set of significant colors to represent an image. This must be necessary to show images in devices which have limited color support or broadcast capacity as hand held devices as PDA's (Personal Digital Assistants), mobile phones, etc. After a color quantization, an image has its color resolution decreased to a specific quantity. This process can reduce the quality of the image if the process were not applied with high precision and specific algorithms. The images produced by the color quantization must be as similar as possible as the original ones which can be evaluated with the use of image fidelity indexes.

Color quantization algorithms can be classified into two classes: splitting algorithms and clustering algorithms. Splitting algorithms divide the color space of an image interactively into disjoint cells according to some criteria until the number of desired cells is reached. Some of the splitting algorithms are: popularity algorithm (Heckbert, 1982), median-cut (Heckbert, 1982), error diffusion (Kang, 1999), Floyd-Steinberg (Kang, 1999), and Stucki (Stucki, 1981). Clustering-based algorithms perform a clustering of the color space into K-desired clusters. The methods involve an initial selection of color map followed by repeatedly updating cluster representatives. C-Means (Parker, 1997) is the most common algorithm of this class.

Algorithms for dithering also work with the reduction of the color space. A review about dithering algorithms is shown in (Alasseur et al, 2003). Dithering is not the best solution for some applications as image processing of historical documents where the background (the paper) must be removed for a character recognition process. A dithering algorithm based on genetic algorithms (GA) and K-Means is proposed in (Freisleben & Schrader, 1997).

The use of computational intelligence for quantization is not new: a technique based on Competitive Hopfield Neural Networks is presented in (Wu et al. 2001). However, this algorithm converges rapidly but it easily finds a local minimum as the solution.

Another problem associated with color quantization is the analysis of the performance of color quantization algorithms. The authors in (Tremeau et al., 1994) define two metrics: LSE (Local Squared Error) and SCAP (Spatial Correlation Among Pixels). Nowadays, however, there are most appropriated metrics as the use of the concepts of image fidelity indexes (Janssen, 2001). We use herein image fidelity metrics in order to evaluate the similarity of the quantized image and the original one.
Thresholding or binarization is a specific type of color quantization that reduces the color palette to just two colors; in general, black and white tones. Some well-known algorithms are: Pun, Kapur, Renyi, Brink, Otsu, Kittler, Percentage of Black and C-Means, for example. Details on these and other algorithms can be found in (Sezgin & Sankur, 2004).

2.2 Image fidelity assessment

Image quality can be defined (Janssen, 2001) "in terms of the satisfaction of two requirements: usefulness (i.e. discriminality of image content) and naturalness (identifiability of image content)". When one has two images and wants to compare them, it is a fidelity value that is searched. This is the main problem of fidelity metrics: the requirement of two images (a reference image and a target one) to make this comparison.
The definition of image fidelity metrics is subject of several studies that come from subjective measures as Mean Opinion Score (MOS) to objective ones as Peak Signal-to-Noise Ratio (PSNR) and Mean Square Error (MSE). Our interest is in objective measures as our work involves sets of thousand of images. PSNR (in dB) is evaluated by:

$$PSNR = 10\log_{10}(\frac{C^2}{MSE}) \tag{1}$$

where C represents the maximum color value (for images).
A fidelity index, Q, is defined in (Wang & Bovik, 2001) in terms of the linear correlation coefficient and the similarities between the mean and variance of two images. This index is defined as:

$$Q = \frac{4.\mu_x.\mu_y.\sigma_{xy}}{(\mu_x{}^2 + \mu_y{}^2).(\sigma_x^2 + \sigma_y^2)} \tag{2}$$

where x and y are the original and tested images respectively, μ_x and μ_y are their means, σ_x and σ_y are their variances and σ_{xy} is the correlation. As defined in (Wang & Bovik, 2001), the range of Q is [-1, 1]. The value of 1 happens when the images are the same (or $y_i = x_i$, for every i). The lowest value, *i.e.* $Q = -1$, occurs when $y_i = 2.\mu_x - x_i$, for every i.

2.3 Genetic algorithms

Genetic algorithms are very useful to solve search problems (Mitchell, 1998), especially for complex, multivariable and non-analytical problems. Therefore, it can be used to solve problems such as identify grayscale levels and the limits between them in a quantization process. This intelligent computing technique is important for the thresholding process proposed herein.
The Genetic Algorithm used in our method follows the flow chart presented in Fig. 2. First, an initial population with P individuals is created. In our method, each individual

represents a set of grayscale levels involved in the process codified in a bit string. For each individual, a simulation tool is run and the fitness function value is returned. Therefore, individuals with the best performance are the stronger ones. For each generation, new individuals are created through crossover processes to compose the next generation. The mutation operator helps to avoid local minimum. The selection operation finds the *P* individuals with higher fitness function and deletes the weakest individuals. At the end of the selection operation, the algorithm checks if the predefined number of generations was reached. The algorithm keeps running until it reaches the limit of generations or it finds a predefined condition.

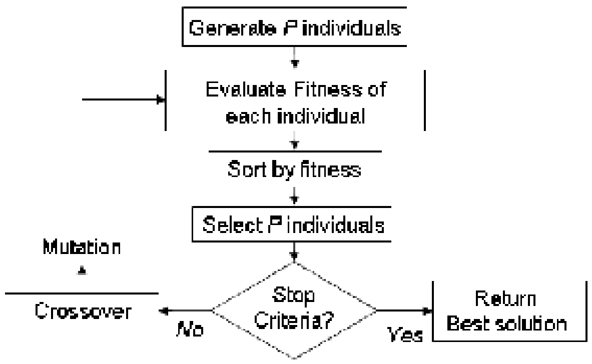


Fig. 2. Flow chart of the genetic algorithm used in our simulations.

The crossover operation is applied to two individuals and it generates two new individuals mixing information of the parents bit strings. Only new individuals are added to the population, clones are discarded. Two individuals in the population have a probability P_c of performing crossover in each generation. In this work we used P_c = 50%.
The mutation operation is applied in a single individual. It consists of complementing bits in the individual string of bits. An individual in the population have a probability P_m of suffering mutation in each generation and, considering mutation in an individual, each bit has a probability of suffering mutation P_{mb}. We used P_m = 5% and P_{mb} = 10% in our simulations.

3. Proposed thresholding algorithm

The main objective of this method is to generate a bi-level image from a historical document. This image shall represent the text as black and background or back-to-front interference as white. So forth, we introduced an intelligent algorithm using genetic algorithm to quantize the original image. The target is to reduce the palette so that the remaining colors (or gray levels) represent classes such as text, background or back-to-front interference. Notice that those classes can be represented by more than one color.
Furthermore, supposing that the text is composed by the darkest grayscale levels, the process is followed by a threshold to classify the pixels as text or non-text, excluding the background and the back-to-front interference. The thresholding divides the classes as text (darkest remaining colors) and non-text.
We used historical documents images stored in 256 gray levels. The information about each gray level that represents a class was codified into a binary bit string. We also codified the

limits that define the threshold in the quantization process. We used 8 bits to represent each color and each limit. Therefore, the individual consists of a bit string with *2z-1* segments, where *z* is the number of gray levels in the novel palette. We used Wang and Bovik's fidelity index (*Q*) as the fitness function. The fitness for an individual is obtained by performing the *Q* factor comparison between the image after the quantization process and the original image. We observed that the *Q* values obtained in our simulation were always in the interval [0, 1]; so this range is considered instead of [-1, 1] as defined by Wang and Bovik. After crossover or mutation, the individual segments shall be sorted according their values.

To illustrate the method we present an example: consider the Figure 1-left as the document to be treated. Figure 1-center shows the best result achieved from binarization and manual exclusion of non desired information. Therefore the target is to define the method to achieve automatically images as showed in Figure 1-center without *a priori* knowledge.

The first step is to execute the quantization based on genetic algorithm, maximizing the *Q* factor.

The second step is to define which of the new gray levels from the quantization should be classified as text. If one considers the image presented in Fig. 1-Left quantized to 4 gray levels, there are three different possibilities to threshold that image: the first one is to classify the darkest grayscale level as text and the others as non text (up ahead classify just the darkest grayscale level will be called *limit1*); the second possibility is to classify the two darkest grayscale levels as text and the others as non text (up ahead classify the two darkest grayscale levels will be called *limit2*). The last possibility is to classify the three darkest grayscale levels as text and the other as non text (from now on this classification will be called *limit3*). Figure 2 shows the resultant images after binarization considering these three limits. The threshold limits were defined as 53, 106 and 137, for limit1, limit2 and limit3, respectively. Visually, the best image was achieved for limit1.

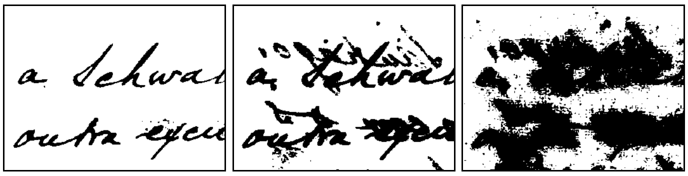


Fig. 3. Thresholding images of the example for binarization with threshold (left) 53, (center) 106 and (right) 137, after the intelligent quantization.

4. Results

It is necessary to determine how many gray levels should be used in the quantization to achieve the best results. For the first step (quantization), the *Q* factor compared with the original grayscale image increases as the number of the gray levels increases. We achieved *Q* = 0.949594, *Q* = 0.967749, *Q* = 0.978475, *Q* = 0.980648, *Q* = 0.985494, *Q* = 0.987862, *Q* = 0.987115, *Q* = 0.99008, *Q* = 0.991768 and *Q* = 0.992203 for 3, 4, 5, 6, 7, 8, 9, 10, 11 and 12 gray levels, respectively. But it must be observed that as our final purpose is binarization as the number of gray levels increases more difficult is to find the better threshold value.

A visual approach is not the best way to quantify the quality of an image. To provide an efficient way to validate the method and to find the optimum point, we compared *Q* factor, PSNR (Peak Signal-to-Noise Ratio) and ROC (Receiver Operating Characteristics) curves of the generated images to the perfect binary image of the example. Figure 3 shows the *Q* factor and PSNR results of the binarized images as a function of the number of grayscale levels in the GA quantization and the number of grayscale levels classified as text (the darkest ones). The best results were obtained for 3 gray levels in the quantization process and limit1 (*Q* = 0.577892 and PSNR = 15.18) and 10 gray levels in the quantization process and limit2 (*Q* = 0.5533 and PSNR = 15.86). The second option presents the best *Q* factor and PSNR. In spite of this, all the three options are quite similar and the first option seems to be more adequate to generalize the data for a group of documents.

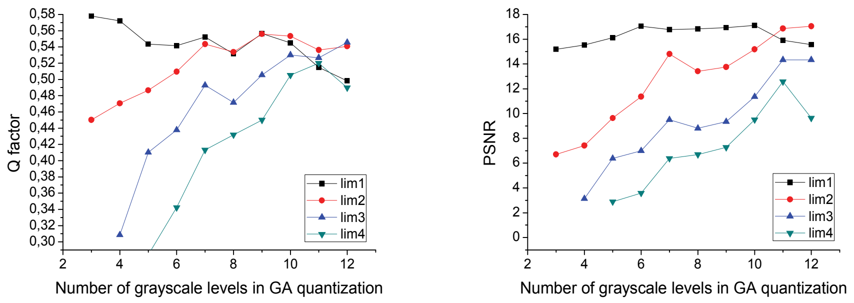


Fig. 3. (left) Q factor and (right) PSNR results as a function of the number of grayscale levels in the GA quantization for each limit case defined.

Another way to evaluate the method is using some measures from Signal Detection Theory (McMillan & Creelman, 2005): precision, recall, accuracy and specificity. Figure 4 presents the plotting of these measures for limit1 and limit2 cases.

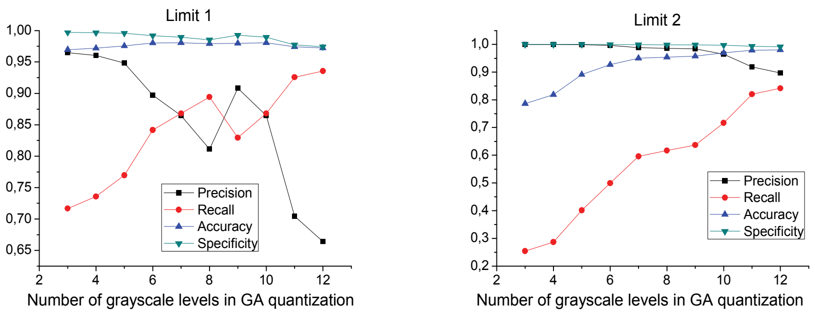


Fig. 4. Precision, recall, accuracy and specificity for (left) Limit 1 and (right) Limit 2 cases as a function of the number of grayscale levels in the GA quantization for each limit case defined.

In order to evaluate the performance of the algorithm using these metrics, a "clean" image was produced for each image in a set of 140 documents. This "clean" image is a bi-level document with only the pixels of the ink. These images were generated manually by visual inspection. With these clean images, we can evaluate the values precision, recall, accuracy and specificity. An efficient algorithm must have these four measures tending to 1.
Table 1 presents the average result for these measures in a comparison between a set of 140 documents binarized by classical algorithms (Sezgin & Sankur, 2004) and the best response

achieved by the new proposed algorithm (images with the higher *Q* value) with their "clean" images. The new algorithm (labeled as GA) achieved high values for all four measures. Table 1 also presents the average values of PSNR and MSE for this set.

Algorithm	Precision	Recall	Accuracy	Specificity	PSNR	MSE
GA	**0.8290**	**0.8273**	**0.9851**	**0.9694**	**21.5167**	**0.0306**
Bernsen	0.839	0.8135	0.9636	0.9847	21.4086	0.0364
Brink	0.8972	0.7176	0.9378	0.9898	20.5486	0.0622
C-Means	0.9656	0.4752	0.7276	0.9919	15.5124	0.2724
daSilva-Lins-Rocha	0.8882	0.7277	0.9459	0.9908	20.3931	0.0541
Fisher	0.9085	0.7274	0.9329	0.9903	20.5406	0.0671
Huang	0.9174	0.6946	0.9146	0.9904	19.8345	0.0854
Johannsen	0.9398	0.6548	0.9317	0.9942	19.3038	0.0683
Kapur	0.0148	0.4073	0.9021	0.9013	16.8976	0.0979
Kittler	0.8587	0.7982	0.9429	0.9847	21.1809	0.0571
Li-Lee	0.872	0.7313	0.9447	0.9899	20.3123	0.0553
Mean Grey Level	0.9116	0.7518	0.9654	0.9891	21.11	0.0346
Niblack	0.862	0.3911	0.8531	0.985	14.6877	0.1469
Otsu	0.8431	0.7128	0.9422	0.9683	19.1462	0.0578
Percentage of Black	0.9902	0.2156	0.6165	0.9985	10.3754	0.3835
Pun	0.941	0.6513	0.9395	0.9957	19.3791	0.0605
Renyi	0.9894	0.1716	0.3683	0.9011	8.6996	0.6317
Sauvola	0.8925	0.1888	0.6045	0.9755	10.0915	0.3955
Iterative Selection	0.3368	0.3356	0.9418	0.9514	18.9542	0.0582
TwoPeaks	0.9402	0.6288	0.9236	0.9947	18.7636	0.0764
White	0.3855	0.757	0.7348	0.9524	15.6284	0.2652
Wu-Lu	0.7792	0.7711	0.9411	0.9595	19.7098	0.0589
Yager	0.7985	0.2502	0.7475	0.9609	12.0418	0.2525
Yen	0.5276	0.8058	0.9358	0.944	19.0102	0.0642

Table 1. Average value of precision (P), recall (R), accuracy (A), specificity (S), PSNR and MSE evaluated by a comparison of bi-level images without noise and the images generated by classical algorithms and the new proposal (labeled GA).

7. Conclusion

It is presented in this paper a technique for automatic thresholding images of historical documents, in special, documents written on both sides of the paper, presenting back-to-front interference. The new method uses genetic algorithms to achieve a quantized image and proceed with a binarization. The resulting images were analyzed using a fidelity index, PSNR and measures from signal detection theory. The method can also be extended to optimize quantization processes for other types of images.

The method proved to be more efficient than several other classical thresholding algorithms in an evaluation using precision, recall, accuracy and specificity.

Currently the bi-level images are being used to improve several steps on an optical character recognition process of these documents such as text segmentation and the recognition *per se*.

8. References

Mello, C.A.B.; Oliveira, A.L.I.; Sanchez, A. Historical Document Image Binarization, *Proceedings of the International Conference on Computer Vision Theory and Applications*, pp. 108-113, ISBN 9789898111210, Funchal, January 2008, INSTICC, Portugal.

Sayood, K. (1996). *Introduction to data Compression*, Morgan Kauffman, ISBN 1558603468, San Francisco.

Parker, J.R. (1997). *Algorithms for Image Processing and Computer Vision*, John Wiley and Sons, ISBN 0471140562, New York.

Heckbert, P. (1982). Color image quantization for frame buffer display. *ACM SIGGRAPH Computer Graphics*, Vol. 16, No. 3, (July 1982), pp. 297-307, ISSN 00978930.

Stucki, P. (1981). MECCA - A multiple error correcting computation algorithm for bilevel image hardcopy reproduction, *Research Report RZ1060*, IBM Research Laboratory, Zurich, Switzerland.

Alasseur, C. ; Constantinides, A.G. ; Husson, L. (2003). Colour Quantisation Through Dithering Techniques, *Proceedings of International Conference on Image Processing*, pp. 469-472, ISBN 0780377516, Barcelona, September 2003, IEEE Press, New Jersey.

Freisleben, B.; Schrader, A. (1997). Color Quantization with a Hybrid Genetic Algorithm, *Proceedings of the International Conference on Image Processing and its Applications*, pp. 86-90, ISBN 085296692X, Ireland, July 1997, IEEE Press, New Jersey.

Wu, Y.; C.Yang; T.Wang. (2001). A New Approach of Color Quantization of Image Based on Neural Networks, *Poceedings of International Joint Conference on Neural Networks*, pp. 973-977, ISBN 0780370465, Washington, USA, July 2001, IEEE Press, New Jersey.

Tremeau, A.; Calonnier, M. ; Laget, B. (1994). Color Quantization Error in Terms of Perceived Image Quality, *Proceedings of the International Conference on Acoustic, Speech and Signal Processing*, pp. V93-V96, ISBN 078031775094, Adelaide, Australia, June 1994, IEEE Press, New Jersey.

Janssen, T.J.W.M. (2001). Understanding image quality, *Proceedings of the International Conference on Image Processing*, pp. 7, ISBN 0780367251, Thessaloniki, Greece, October 2001, IEEE Press, New Jersey.

Sezgin, M.; Sankur, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation, *Journal of Electronic Imaging*, Vol. 1, No.13, (January 2004), pp. 146-165, ISSN 10179909.

Wang, Z.; Bovik, A.C. (2002). A Universal Image Quality Index. *IEEE Signal Processing Letters*, Vol. 9, No. 3, (March 2002), pp. 81-84, ISSN 10709908.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*, MIT Press, ISBN 0262631857, Cambridge.

McMillan, N.A.; Creelman, C.D. (2005). *Detection Theory*. LEA Publishing, ISBN 0805842306, New Jersey.

Segmentation of Greek Texts by Dynamic Programming

Pavlina Fragkou[1], Athanassios Kehagias[2] and Vassilios Petridis[1]
[1]Department of Electrical and Computer Engineering,
[2]Department of Math., Phys., and Computer Sciences,
Faculty of Engineering, Aristotle University of Thessaloniki,
Greece

1. Introduction

In this paper we present an approach for the segmentation of concatenated texts. The text segmentation problem can be stated as follows: given a text which consists of several parts (each part dealing with a different subject) it is required to find the boundaries between the parts. In other words, the goal is to divide a text into homogeneous segments so that each segment deals with a particular subject while contiguous segments deal with different subjects. In this manner, documents relevant to a query can be retrieved from a large database of unformatted (or loosely formatted) text. The problem appears often in information retrieval and text processing.

Our approach combines elements from several previously published text segmentation algorithms and achieves a significant improvement in segmentation accuracy by following a supervised approach. More specifically, we perform linear segmentation of concatenated texts by minimizing a segmentation cost which consists of two parts: (a) within-segment word similarity (expressed in terms of dotplot density) and (b) prior information about segment length. The minimization is effected by dynamic programming, which guarantees that the globally optimal segmentation is obtained. We are concerned with linear text segmentation, which should be distinguished from hierarchical text segmentation (Yaari, 1997; Yaari, 1999); the latter attempts to find a tree-like structure in the text segments, while linear segmentation is based on the assumption that text has a linear structure thus segments appear in sequential "flat" order. Let us note that hierarchical segmentation is perhaps more appropriate for discourse segmentation because it creates a hierarchy of all topics discussed. Every sub-topic is appropriately related to the topic with which is related to in a deeper level placed in a form of "leaf".

Our method has successfully applied to Greek texts proving to be very innovating and promising. Results regarding segmentation of English texts can be found in (Kehagias et al., 2004(a); Kehagias et al., 2004(b)). The remainder of the paper is organized as follows: in Section 2 we present research approaches on the area of text segmentation, in Section 3 we introduce our algorithm, in Section 4 we present experiments to evaluate the algorithm. Finally, in Section 5 we discuss our results.

2. Related work

Text segmentation approaches are based in the theory of Halliday and Hasan (Halliday & Hasan, 1976) according to which, each text is described by two complementing elements: *cohesion* and *coherence*. *Cohesion* is described as the quality property of a text and is detected by the simultaneous appearance of semantically similar words. Cohesion is present when an element in the text is best interpreted in light of a previously (or rarely a subsequent) element within the text. *Coherence* on the other hand holds between two tokens in the text which are either of the same type or are semantically related in a particular way (such as a word or group of words having a clearly definable relationship with a previously used word i.e. belonging to the same theme or topic). According to Halliday and Hasan semantic coherence and cohesion are identified by the following five semantic relations: (1) repetition with similarity, (2) repetition without similarity (3) repetition through reference to a higher category in which the aforementioned word entity belongs to (4) systematic semantic relationship (5) non- systematic semantic relationship. In the same spirit, Raskin and Weiser (Raskin & Weiser, 1987) defined as a criterion for cohesion and coherence word repetition and comparative apposition, where the first focus on word repetition or synonyms of them and the latter on words that present the tendency to co-occur within a document.

In this paper, the focus is stressed towards (concatenated) text segmentation, which is often distinguished from discourse segmentation. The goal of discourse segmentation is to split a single large text into its constituent parts (e.g. to segment an article into sections); this problem is addressed, for instance, in (Hearst, 1994; Hearst & Plaunt, 1993; Heinonen, 1998; Yaari, 1997; Yaari, 1999). On the other hand, the goal of (concatenated) text segmentation is to split a stream of independent, concatenated texts (e.g. to segment a transcript of news into separate stories); this problem is addressed, for example, in (Beeferman et al., 1999; Choi, 2000; Choi et al., 2001; Ponte & Croft, 1997; Reynar, 1994; Reynar & Ratnaparkhi, 1997; Utiyama & Isahara, 2001). The two problems are similar but not identical; our algorithm could conceivably be applied to discourse segmentation, but our main interest is in concatenated text segmentation and all the experiments we present here fall into this category.

Generally speaking, text segmentation is a two step procedure. The first step involves the calculation of segment *homogeneity* while the second the identification of segment boundaries. The calculation of segment *homogeneity* (or alternatively *heterogeneity*) performed by methods appearing in the literature presents a strong variation. On the one hand, a family of methods makes use of linguistic criteria such as cue phrases, punctuation marks, prosodic features, reference, syntax and lexical attraction (Beeferman et al. (1997), Hirschberg & Litman (1993), Passoneau & Litman (1993)). On the other hand the second family, following Halliday and Hasan's theory (Halliday & Hasan (1976)), utilizes statistical similarity measures such as word co-occurrence. Roughly speaking, two parts of the text are considered similar if they have many words in common. This is a popular approach, according to which parts of a text having similar vocabulary are likely to belong to a coherent topic segment. For example the linear discourse segmentation algorithm proposed by Morris and Hirst (Morris & Hirst (1991)) is based on *lexical cohesion relations* determined by use of Roget's thesaurus (Roget (1977)). In the same direction Kozima's algorithm (Kozima (1993), Kozima & Furugori (1993)) computes the semantic similarity between words using a semantic network constructed from a subset of the Longman Dictionary of Contemporary English. Local minima of the similarity scores correspond to the positions of topic boundaries in the text. Other authors have used fairly sophisticated word co-occurrence statistics such as LSA, LCA, ranking etc. Choi, 2000; Choi et al., 2001; Hearst, 1994; Hearst & Plaunt, 1993; Utiyama & Isahara, 2001).

The identification of segment boundaries usually requires the minimization of a segmentation cost function. An efficient way to perform this is by the use of techniques such as dynamic programming. This is due to the fact that dynamic programming is based on the intuition that a longer problem can be solved by properly combining the solution to various sub-problems. For example, consider the sequence or "path" of transformed words that comprise the minimum edit distance between the strings "intention" and "exention". Imagine one string (perhaps it is exention) that is in this optimal path (whatever it is). The intuition of dynamic programming is that if exention is in the optimal operation list, then the optimal sequence must also include the optional path from intention to exention. This is because, if there were a shorter path from intention to exention then we could use it instead, resulting in the shortest path and the optimal sequence wouldn't be optimal, thus leading to contradiction. Another benefit of dynamic programming is that at every point of execution the optimal solution from the previously examined observations was calculated avoiding thus backtracking (Berteskas, 1987). This approach has been used in the past (in Heinonen, 1998; Ponte & Croft, 1997; Xiang & Hongyuan, 2003) and also, implicitly, in (Utiyama & Isahara, 2001). Other authors do not cast segmentation as a formal optimization problem; rather they construct a similarity matrix which they segment using divisive clustering, which can be considered as a form of approximate and local optimization (Choi, 2000; Choi et al., 2001; Reynar, 1994; Reynar & Ratnaparkhi, 1997; Yaari, 1997; Yaari, 1999).

As we have already mentioned, we formulate segmentation as the minimization of a segmentation cost which depends on within-segment homogeneity and deviation from expected segment length. We measure within-segment homogeneity by word co-occurrence by operating at the sentence level and consider two sentences to be similar if they have even a single word in common. We use a "global" similarity comparison, i.e. we evaluate the similarity between all parts of a text (for example between every pair of sentences that appear in the text, even if they are not adjacent to each other). This approach is used by several authors (Choi, 2000; Choi et al., 2001; Ponte & Croft, 1997; Reynar, 1994; Reynar & Ratnaparkhi, 1997; Xiang & Hongyuan, 2003), but it should be noted that "local" comparison (i.e. only between adjacent sentences) has also been used in the past (Hearst, 1994; Hearst & Plaunt, 1993; Heinonen, 1998). To penalize deviations from the expected segment length we use a "length-model"; this approach has been used in the past by several authors (Heinonen, 1998; Ponte & Croft, 1997). We find the globally minimal segmentation cost by dynamic programming.

Current approaches to text segmentation include an improvement of the dotplotting technique (Ye et al., 2005) introduced by Reynar (Reynar, 2004), an improvement of Latent Semantic Analysis for text segmentation (Bestgen, 2006), a model of text segmentation based on ideas from multilabel classification for segmenting sentences into tokens (McDonald et al., 2005) as well as a novel parameter-free unsupervised text segmentation method, which is formulated as (variational) Bayes estimation of an HMM from an input text stream (Koshinaka et al., 2005). Teo Yung Kiat' master thesis present an attempt to extend and improve our method (Kiat, 2005). Advances to topic segmentation (closely related to text segmentation) include methods performing topic segmentation method based on weighted lexical chains (Sitbon & Bellot, 2005), as well as a new informative similarity measure based on word co-occurrences (Dias & Alves, 2005).

It is worth mentioning that text segmentation is widely used in other closely related scientific areas such as speech segmentation i.e. to identify breaks and discourse boundaries by expert and/or naive listeners (Auran et al., 2005), spoken multiparty dialogue and tutorial dialogue segmentation (Olney & Cai, 2005; Hsueh et al., 2006). Text segmentation techniques are also applied to entity extraction and noun-phrase chunking (Ursu et al., 2005) as well as to semantic annotation of transcripts of television news broadcasts produced through automatic speech recognition (ASR) (Dowman et al., 2005). Text segmentation proves to be beneficial in a number of scientific areas such as corpus linguistics, discourse psychology and even education. This is due to the fact that text segmentation is based on topic change. Topic change or topic coherence is highly related to the vocabulary used by each author, the subconscious mechanism of language variation, the part of speech of words that he/she uses which may reveal positivity, sociability, complexity or negativity, self concern emphasis and implicitness. In psychological perspective, text segmentation may reveal if an author express its subject in question by following a coherence and progressive apposition of his arguments or it interrupts his argumentation by making references to less important or even non relevant subjects. Thus, text segmentation can be found useful in studies concerning topic and authorship attribution where topic change can highly be related to the vocabulary used by each author (Stamatatos et al., 2001). Finally, text segmentation can easily be applied as a preliminary step to text summarization.

3. Method and algorithm

3.1 Text representation

A text consists of words which are organized in sentences. We assume that sentence boundaries are correctly marked in the text. Hence we will assume from now on that the basic text unit is the sentence and that segment boundaries occur only at the end of sentences. Consider a text which contains T sentences and L distinct words (i.e. a vocabulary of size L). We define a T x T similarity matrix D as follows (s, t = 1, 2,…, T)

$$D_{s,t} = \begin{Bmatrix} 1, \textit{if sentences s and t have at least a common word and } s \neq t \\ 0, \textit{otherwise} \end{Bmatrix} \quad (1)$$

It is worth mentioning that, by the term "words" we mean any word used by the author of that segment but not its grammatical form. In our study we do not perform an in depth linguistic process i.e. grammatical parsing and co-reference resolution in order to discover the context under which each word appears or the sequence of appearance of words. Our research is based on the hypothesis that each segment corresponds to a different topic. The description of that topic tends to be performed by using a small number of characteristic words that belong to a limited size vocabulary. On the other hand, highly informative words tend to appear more that one times, thus, the importance of them is reinforced in the similarity matrix. Finally, it is worth mentioning that, none of the algorithms dealing with the same problem make use of grammatical items. An opposite approach would lead to a misleading comparison of obtained results. Additionally, it is our belief that, it is the choice of words that the authors use in order to express their topic than the grammatical property of those that it acts as a discriminative factor in the topic i.e. segment change identification. Lastly, we believe that in case where high informative combination of words i.e. n-grams appear in the segment, the fact that the information that they contain is represented not as a whole but with their consisting words as individuals does not lead to "loss" of the information contained.

Hence, if $D_{s,t} = 1$ we assume that the s-th and t-th sentence are similar. Figure 1 provides the dotplot (Choi, 2000; Choi et al., 2001; Reynar, 1994; Reynar & Ratnaparkhi, 1997) of a D matrix corresponding to a 91-sentences text; black squares correspond to 1's and white squares to 0's. Consecutive groups of sentences which have many words in common appears as submatrices of D with many 1's; in Figure 1 they appear as high density squares. Candidate segments appear, for example, between sentences 11 and 18, 41 and 52 etc. Hence the dotplot gives a visual representation of the structure of the text.

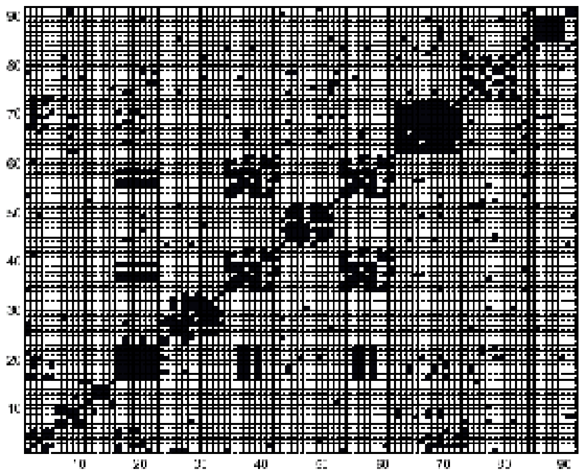


Figure 1: The similarity matrix D corresponding to a text containing 91 sentences, hence D is a 91 x 91 matrix. A black dot at position (s, t) indicates that the s-th and t-th sentence have at least one word in common

It is worth mentioning that, the total number of shared words is indirectly depicted in the dotplot similarity matrix. Sentences that have an important number of shared words lead to regions containing a lot of '1's. Sentence length is not considered here, as it would require the calculation of the total number of words belonging to each sentence, the number of common and non common words between sentences as well as sentence length normalization. Such approach is left for future research.

3.2 Segmentation cost

A segmentation is a partition of the set {1,2,…,T} into K subsets (i.e. segments) of the form {1,2,.., t_1}, {t_1+1, t_1+2,…, t_2}, ..., {$t_{K-1}+1$, $t_{K-1}+2$,…,T}(where K is a variable number and K ≤ T). A more economical description of the segmentation is given by a (variable length) vector t = $(t_0, t_1, \ldots, t_K)$, where, t_0, t_1, …., t_K are the segment boundaries which satisfy $0 = t_0, < t_1 < \ldots < t_K - 1 < t_K = T$.

We now introduce a "segmentation cost" function J(t): for every segmentation t, J(t) returns a real number; J(t) will be designed in such a way that it achieves small values when t designates high-density submatrices of D. We start with the function

$$\mathrm{J}_0(t) = \frac{\sum_{s=t_{k-1}+1}^{t_k} \sum_{t=t_{k-1}+1}^{t_k} D_{s,t}}{\left(t_k - t_{k-1}\right)^r} \tag{2}$$

which can be interpreted as follows. The numerator is the total number of 1's contained into the D submatrix which corresponds to the k-th segment $\{\mathrm{t}_{k-1}+1, \mathrm{t}_{k-1}+2, \ldots, \mathrm{t}_k\}$. When the parameter r=2, the denominator $\left(\mathrm{t}_k - \mathrm{t}_{k-1}\right)^r$ corresponds to the area of the sub-matrix and $\mathrm{J}_0(t)$ is the "segment density". In the case r≠2, $\mathrm{J}_0(t)$ corresponds to a "generalized density" which balances the degree of influence of the surface with regard to the "information" (i.e. the number of 1's) included in it. A "good" segmentation t is characterized by large values of $\mathrm{J}_0(t)$, which indicate strong within-segment similarity.

In many cases some information will be available regarding the expected segment length; for instance we may use training data to estimate its mean value μ and standard deviation σ. We incorporate this information into a function:

$$\mathrm{J}_1(t) = \sum_{k=1}^{K} \frac{\left(t_k - t_{k-1} - \mu\right)^2}{2 \bullet \sigma^2} \tag{3}$$

A "good" segmentation t is characterized by small values of $\mathrm{J}_1(t)$, which indicate small deviation from the expected segment length (1).

Finally, we form J by a weighted combination of J_0 and J_1:

$$\mathrm{J}(\mathrm{t}; \mu, \sigma, \mathrm{r}, \gamma) = \gamma \bullet \mathrm{J}_1(t) - (1-\gamma) \bullet \mathrm{J}_0(t) = \sum_{k=1}^{K} \left[\gamma \bullet \frac{\left(t_k - t_{k-1} - \mu\right)^2}{2 \bullet \sigma^2} \right] - (1-\gamma) \bullet \frac{\sum_{s=t_{k-1}+1}^{t_k} \sum_{t=t_{k-1}+1}^{t_k} D_{s,t}}{\left(t_k - t_{k-1}\right)^r} \tag{4}$$

where we stress the dependence of J on the parameters, μ, σ, r and γ.

3.3 Minimization by dynamic programming

A "good" segmentation vector t yields a small value of the corresponding $\mathrm{J}(\mathrm{t}; \mu, \sigma, \mathrm{r}, \gamma)$ (i.e. segments with high density and small deviation from average segment length). The optimal segmentation $\hat{t}$ is the one which yields the global minimum of $\mathrm{J}(\mathrm{t}; \mu, \sigma, \mathrm{r}, \gamma)$; note that $\hat{t}$ specifies not only the optimal positions of the segment boundaries $\mathrm{t}_0, \mathrm{t}_1, \ldots, \mathrm{t}_K$ but also the optimal number of segments K; in other words, our algorithm automatically determines the optimal K.

[1] Many other functional forms can be used for $\mathrm{J}_1(t)$; in Kehagias et al., 2004(a) and Kehagias et al., 2004(b), we have explored some alternatives but we have found that the form used here gives the best results.

Our J(t; μ, σ, r, γ) has an additive form which is well suited for the global minimization by dynamic programming. The following algorithm implements the basic dynamic programming idea (for a detailed justification the reader can consult (Bertsekas, 1987)).

Dynamic Programming for Text Segmentation

Input: The T x T similarity matrix D; the parameters μ , σ , r, γ :

```
Initialization
For t = 1, 2,…,T
q = 0
    For s = 1,2, ..., t -1
        q = q+ $D_{s,t}$
        $S_{s+1,t} = \frac{q}{(t-s)^r}$
    End
End
Minimization
$C_0 = 0, Z_0 = 0$
For t = 1, 2,…,T
$C_t = \infty$
    For s = 0, 1, ..., t -1
        If
            $C_s + \gamma \bullet \frac{(t-s-\mu)^2}{2 \bullet \sigma^2} - (1-\gamma) \bullet S_{s+1,t} \le C_t$
        Then
            $C_t = C_s + \gamma \bullet \frac{(t-s-\mu)^2}{2 \bullet \sigma^2} - (1-\gamma) \bullet S_{s+1,t}$
            $Z_t = s$
        EndIf
    End
End
BackTracking
K = 0, $s_K = T$
While $Z_{s_K} > 0$
    K = K + 1
    $s_K = Z_{s_{K-1}}$
End
K = K + 1, $s_K = 0$ , $\hat{t}_0 = 0$
For k = 1, 2,…,K
    $\hat{t}_k = s_{K-k}$
End
```

Output: The optimal segmentation vector $\hat{t} = (\hat{t}_0, \hat{t}_1, \dots, \hat{t}_K)$.

Upon completion of the minimization part of the algorithm we have computed the optimal segmentation cost for sentences 1 until T, i.e. for the entire text. The backtracking part first creates the sequence $s_0, s_1, .., s_K$ which are the optimal segment boundaries in reverse order and then reverses this sequence to produce the optimal $\hat{t} = (\hat{t}_0, \hat{t}_1, \dots, \hat{t}_K)$. Note that K, the optimal number of segments is computed automatically.

4. Experiments - results

In this section we present the experiments we conducted to evaluate our algorithm. We evaluate the algorithm using the following three indices: *Precision, Recall* and Beeferman's P_k metric (Beeferman et al., 1999). *Precision* is defined as "the number of the estimated segment boundaries which are actual segment boundaries" divided by "the number of the estimated segment boundaries". *Recall* is defined as "the number of the estimated segment boundaries which are actual segment boundaries" divided by "the number of the true segment boundaries". It is worth mentioning that the F measure, which combines the results of Precision and Recall, is not used here, due to the fact that both Precision and Recall penalize equally segment boundaries that are "close" to the actual i.e. true boundaries with those that are less close to the true boundary. For that reason, Beeferman proposed an new metric P_k which measures segmentation inaccuracy; intuitively, P_k measures the proportion of "sentences which are wrongly predicted to belong to different segments (while actually they belong to the same segment)" or "sentences which are wrongly predicted to belong to the same segment (while actually they belong in different segments)" (for a precise definition of P_k see (Beeferman et al., 1999).

The variation of the P_k measure named WindowDiff index which was proposed by Pevzner and Hearst (Pevzer & Hearst, 2002) and remedies several problems of the P_k measure is not used in this paper due to the number of experiments conducted and the fact that already published results used for comparison are only reported in terms of P_k .

While several papers regarding the segmentation of English texts have appeared in the literature, we are not aware of any similar work regarding Greek texts. Furthermore, because Greek is a highly inflected language (much more than English) the segmentation problem is harder for Greek, as will be explained in the following. Hence some enhancements to the basic segmentation algorithm are required.

In the sequel we present experiments which use a Greek text collection compiled from Stamatatos'corpus 2(Stamatatos et al., 2001) comprising of text downloaded from the website http://tovima.dolnet.gr of the newspaper entitled 'To Vima'. This newspaper contains articles belonging to one of the following categories: 1) Editorial, diaries, reportage, politics, international affairs, sport reviews 2) cultural supplement 3) Review magazine 4) Business, finance 5) Personal Finance 6) Issue of the week 7) Book review supplement 8) Art review supplement 9) Travel supplement. Stamatatos et al. (Stamatatos et al., 2001)

2 The authors would like to thank professor E. Stamatatos for providing us the corpus of Greek articles.

constructed a corpus collecting texts from supplement no. 2) which includes essays on science, culture, history etc. Stamatatos et al. selected 10 authors and used 30 texts per author. They didn't perform any manual text preprocessing or text sampling; however, they removed all the unnecessary heading irrelevant to the text itself. In order to minimize the potential change of the personal style of an author over time, they chose to download texts taken from the issues published from 1997 till early 1999. The thematic areas of each author are shown in Table 1.

Due to the nature of the newspaper supplement, texts included in, undergo some low-level post editing -as opposed to editorial or reportage articles, which are subject to a stricter editing- so that they conform to the overall style of the newspaper. Therefore, the style of the specific authors is more personal and independent of outer influences. An example of those documents is listed in Appendix B.

Author	**Thematic Area**
Alachiotis	Biology
Babiniotis	Linguistics
Dertilis	History, Society
Kiosse	Archeology
Liakos	History, Society
Maronitis	Culture, Society
Ploritis	Culture, History
Tassios	Technology, Society
Tsukalas	International Affairs
Vokos	Philosophy

Table 1. List of Authors and their Thematic Areas in the Stamatatos's collection of Greek texts.

We created several texts, each consisting of segments by various authors. Each author is characterized by her/his vocabulary hence our goal is to segment the text into the parts written by the various authors. Before creating the actual texts, some preprocessing (performed in a totally automatic manner) of the Stamatatos collection was necessary. Because Greek is a heavily inflected language, a word may appear in many different forms. Then, if one considers each inflected form as a separate element of the vocabulary, the result is a larger vocabulary, which considerably complicates the segmentation problem. To address this issue, we must identify various inflected forms as belonging to the same word; but for Greek this cannot be done using a simple approach such as stemming. Instead, we used the POS tagger developed by Orphanos et al. (see Orphanos & Christodoulakis, 1999; Orphanos & Tsalidis,1999) and the Appendix A, 3) to substitute each word by a "canonical", lemmatized form. More specifically, at the first stage, punctuation marks and numbers were removed as well as all words that aren't either nouns, verbs, adjectives or adverbs (the stop list used here is very similar to the one used for English texts). After that, every remaining word in the text was substituted by its lemma, determined by the tagger. In case the tagger could not find the lemma of a particular word (usually this happened because the word was

[3] The authors would like to thank professor G. Orphanos for kindly letting us use the POS Tagger.

not contained in the tagger Lexicon) no substitution was made and the word was kept in the form appearing in the text. We also kept the information regarding sentence ends.
We present two groups of experiments, which differ in the length of segments created and the number of authors used for the creation of the texts to segment.

4.1 Experiment group 1

The collection of texts used for the first group of experiments consists of 6 datasets: Set0,..., Set5. Each of those datasets differ in the number of authors used for the generation of the texts to segment and consequently in the number of texts used from the entire collection, as listed in Table 2.
For each of the above datasets, we constructed four subsets, which differ in the number of the sentences appearing in each segment. Let L_{min} and L_{max} be the smallest and largest number of sentences which a segment may contain. We have used four different (L_{min}, L_{max}) pairs: (3,11), (3,5), (6,8) and (9,11). Hence Set0 contains 4 subsets: Set01, Set02, Set03 and similarly for Set1, Set2, ..., Set5. The datasets Set*1 are the ones with (L_{min}, L_{max}) = (3,11), the datasets Set*2 are the ones with (L_{min}, L_{max}) =(3,5), and so on. Let also $\{X_1,...,X_n\}$be the authors contributing to the generation of the dataset. We generated the texts in the dataset by the following procedure.
Each text is the concatenation of ten segments. For each segment we do the following.

1. We select randomly an author from $\{X_1,...,X_n\}$. Let I be the selected author.
2. We select randomly a text among the 30 available that belong to the I author. Let k be the selected text of author I.
3. We select randomly a number $l \in (L_{min}, L_{max})$.
4. We extract l consecutive lines from text k (starting at the first sentence of the text). Those sentences constitute the generated segment.

Once we have created a dataset, we split it into a training set and a test set, we use the training data to compute μ, σ and optimal γ and r values (by the validation procedure explained in the sequel) and finally run our algorithm on the test data.

Dataset	Authors	No. of docs per set
Set0	Kiosse, Alachiotis	60
Set1	Kiosse, Maronitis	60
Set2	Kiosse, Alachiotis, Maronitis	90
Set3	Kiosse, Alachiotis, Maronitis, Ploritis	120
Set4	Kiosse, Alachiotis, Maronitis, Ploritis, Vokos	150
Dataset	All Authors	300

Table 2. List of the sets complied in the 1st group of experiments using Greek texts and the author's texts used for each of those.

Recall that the segmentation algorithm uses four parameters: μ , σ , r and γ . As already mentioned μ and σ can be interpreted as the average and standard deviation of segment length; it is not immediately obvious how to choose values for r and γ . We use training data and a parameter validation procedure to determine appropriate μ , σ , r and γ values; then we evaluate the algorithm on (previously unseen) test data. More specifically:

1. We choose randomly half of the texts in the dataset to be used as training texts; the rest of the samples are set aside to be used as test texts.
2. We determine appropriate μ and σ values using all the training texts and the standard statistical estimators.
3. We determine appropriate r and γ values by running (on the training texts) the segmentation algorithm with 80 possible combinations of r and γ values; namely we let γ take the 20 values 0.00, 0.01, 0.02, ... , 0.09, 0.1, 0.2, 0.3, ... , 1.0 and let r take the values 0.33, 0.5, 0.66, 1. The optimal (γ , r) combination is the one which yields the minimum P_k value.
4. We apply the algorithm to the test texts using previously estimated μ , σ , r and γ values.

The aforementioned procedure is repeated five times for all sets; the resulting values of Precision, Recall and P_k are averaged. This is performed in order to avoid any problems that can arise from the fact that the various sets of corpus are composed of many segments repeatedly drawn from a small number of different texts. Moreover the fact that texts consisting the training and testing set are randomly selected and the aforementioned procedure is repeated five times, minimizes the probability that a (probably) significant part of the training and testing set is in fact in common. Even this was the case the remaining not common texts would act as "negative" examples i.e. as far as the calculation of the mean and standard deviation is concerned.
In Table 3 we give the values of Precision, Recall and P_k obtained by our algorithm. We also run Choi's and Utiyama's algorithms on the same task; the results are given in Tables 4 and 5.
In Tables 6, 7 and 8 we give the same results averaged over all datasets which have segments of same length. It can be seen that in all cases our algorithm performs significantly better than both Choi's and Utiyama's algorithms. Let us note that the best performance has been achieved for γ in the range [0.08, 0.4] and for r equal to either 0.5 or 0.66.

4.2 Experiment group 2

The second group of experiments also uses Stamatatos's collection; however, the texts are generated using a somewhat different procedure. We constructed a single dataset which contains 200 texts, with every author represented (in other words, the author set is always $\{x_1, x_2, ..., x_{10}\}$). Each text is the concatenation of ten segments. For each segment we do the following:

1. We select randomly an author from $\{x_1, x_2, ..., x_{10}\}$. Let I be the selected author.
2. We select randomly a text among the 30 available that belong to the I author. Let k be the selected text of author I. The selected text is read and scanned in order to determine the number of paragraphs it contains. Let Z be the number of paragraphs that k-th text contains.
3. We select randomly a number p $\in \{1, ..., Z\}$ corresponding to the number of paragraphs that the generated segment will contain.
4. We select randomly a number m $\in \{1, ..., Z - p\}$ corresponding to the "starting paragraph". Thus the segment contains all the paragraphs of text k starting from paragraph m and ending at the paragraph m + p.

The procedure described above gives texts which are longer than the ones used in Experiment Group 1. Hence the segmentation task in the current group of experiments segmentation of such texts is more difficult than the previous one. Table 9 lists the results we obtained using our algorithm and the ones by Choi and Utiyama. It can be seen again that our algorithm performs better than both Choi's and Utiyama's algorithms.

Dataset	**Precision**	**Recall**	P_k	**Dataset**	**Precision**	**Recall**	P_k
Set01(3-11)	70.65%	71.11%	14.04%	Set31 (3-11)	59.99%	58.67%	17.93%
Set02 (3-5)	86.82%	87.11%	6.20%	Set32 (3-5)	84.44%	83.56%	7.36%
Set03 (6-8)	96.44%	96.44%	0.82%	Set33 (6-8)	86.22%	86.22%	3.28%
Set04(9-11)	93.33%	93.33%	0.84%	Set34 (9-11)	91.11%	91.11%	1.45%
Set11(3-11)	63.86%	67.11%	15.82%	Set41 (3-11)	57.99%	51.11%	17.38%
Set12 (3-5)	82.98%	83.56%	8.47%	Set42 (3-5)	85.00%	84.89%	6.76%
Set13 (6-8)	91.11%	91.11%	2.81%	Set43 (6-8)	88.89%	88.89%	2.65%
Set14(9-11)	94.67%	94.67%	0.98%	Set44 (9-11)	91.11%	91.11%	1.39%
Set21(3-11)	71.14%	60.89%	14.42%	Set51 (3-11)	65.74%	61.78%	14.54%
Set22 (3-5)	90.00%	89.78%	3.45%	Set52 (3-5)	81.56%	81.78%	6.49%
Set23 (6-8)	91.11%	91.11%	2.15%	Set53 (6-8)	89.33%	89.33%	3.57%
Set24(9-11)	92.44	92.44	1.25%	Set54 (9-11)	88.89%	88.89%	1.86%

Table 3. The precision, recall and P_k values obtained by our algorithm for the 1st group of experiments using Greek texts.

Dataset	**Precision**	**Recall**	P_k	**Dataset**	**Precision**	**Recall**	P_k
Set01 (3-11)	65.75%	65.75%	17.06%	Set31 (3-11)	57.75%	57.75%	20.38%
Set02 (3-5)	74.50%	74.50%	16.68%	Set32 (3-5)	70.75%	70.75%	17.40%
Set03 (6-8)	76.50%	76.50%	11.72%	Set33 (6-8)	62.00%	62.00%	17.12%
Set04 (9-11)	64.75%	64.75%	15.08%	Set34 (9-11)	62.00%	62.00%	16.10%
Set11 (3-11)	67.50%	67.50%	16.91%	Set41 (3-11)	57.50%	57.50%	17.38%
Set12 (3-5)	67.75%	67.75%	19.23%	Set42 (3-5)	73.25%	73.25%	15.76%
Set13 (6-8)	72.50%	72.50%	14.74%	Set43 (6-8)	62.50%	62.50%	17.41%
Set14 (9-11)	68.25%	68.25%	14.00%	Set44 (9-11)	63.75%	63.75%	13.70%
Set21 (3-11)	61.00%	61.00%	19.93%	Set51 (3-11)	60.36%	60.50%	17.63%
Set22 (3-5)	73.50%	73.50%	16.15%	Set52 (3-5)	70.50%	70.50%	16.39%
Set23 (6-8)	69.00%	69.00%	15.40%	Set53 (6-8)	67.25%	67.25%	15.85%
Set24 (9-11)	71.75%	71.75%	12.26%	Set54 (9-11)	70.00%	70.00%	12.43%

Table 4. The precision, recall and P_k values obtained by Choi's algorithm for the 1st group of experiments using Greek texts.

Dataset	Precision	Recall	P_k	Dataset	Precision	Recall	P_k
Set01 (3-11)	69.94%	65.55%	15.33%	Set31 (3-11)	61.25%	58.44%	17.64%
Set02 (3-5)	74.16%	59.11%	19.99%	Set32 (3-5)	66.45%	52.88%	20.98%
Set03 (6-8)	80.60%	76.88%	8.94%	Set33 (6-8)	71.88%	70.66%	11.80%
Set04 (9-11)	76.18%	74.45%	8.84%	Set34 (9-11)	67.60%	71.78%	8.75%
Set11 (3-11)	71.41%	68.44%	14.99%	Set41(3-11)	57.77%	56.44%	20.61%
Set12 (3-5)	74.75%	59.11%	18.70%	Set42 (3-5)	71.25%	56.22%	20.07%
Set13 (6-8)	84.77%	83.33%	7.08%	Set43 (6-8)	67.96%	66.44%	12.64%
Set14 (9-11)	81.71%	79.11%	9.10%	Set44 (9-11)	70.23%	72.88%	8.50%
Set21 (3-11)	63.59%	61.11%	18.26%	Set51 (3-11)	60.00%	56.61%	17.41%
Set22 (3-5)	70.57%	53.33%	21.51%	Set52 (3-5)	62.83%	47.55%	23.51%
Set23 (6-8)	77.73%	74.00%	10.75%	Set53 (6-8)	69.56%	66.89%	13.84%
Set24 (9-11)	74.53%	77.33%	7.80%	Set54 (9-11)	68.55%	70.22%	9.99%

Table 5. The precision, recall and P_k values obtained by Utiyama and Isahara's algorithm for the 1st group of experiments using Greek texts.

Dataset	Precision	Recall	P_k
Set*1 (3-11)	64.90%	61.77%	15.69%
Set*2 (3-5)	85.13%	85.11%	6.45%
Set*3 (6-8)	90.51%	90.51%	2.54%
Set*4 (9-11)	91.92%	91.92%	1.29%

Table 6. The precision, recall and P_k values obtained by our algorithm for the 1st group of experiments using Greek texts, averaged over datasets with same-length segments.

Dataset	Precision	Recall	P_k
Set*1 (3-11)	61.64%	61.66%	18.43%
Set*2(3-5)	71.70%	71.70%	16.93%
Set*3 (6-8)	68.29%	68.29%	15.37%
Set*4 (9-11)	66.75%	66.75%	13.93%

Table 7. The precision, recall and P_k values obtained by Choi's algorithm for the 1st group of experiments using Greek texts, averaged over datasets with same-length segments

Dataset	Precision	Recall	P_k
Set*1 (3-11)	64.00%	61.10%	17.37%
Set*2 (3-5)	70.00%	54.70%	20.79%
Set*3 (6-8)	75.42%	73.03%	10.84%
Set*4 (9-11)	73.13%	74.29%	8.83%

Table 8. The precision, recall and P_k values obtained by Utiyama and Isahara's algorithm for the 1st group of experiments using Greek texts, averaged over datasets with same-length segments.

Algorithm	**Precision**	**Recall**	P_k
Ours	60.60%	57.00%	11.07%
Choi	44.62%	44.62%	19.44%
Utiyama	56.76%	67.22%	12.28%

Table 9. The precision, recall and P_k values for the 2nd group of experiments using Greek texts.

It is worth mentioning that, the experiments were conducted in a Pentium III 600 MHz with 256 Mbyte RAM memory. The training time of each group was calculated and proved that it is less than two minutes. The average time of calculation for the segmentation of a text by our algorithm was 0.91 seconds.

5. Conclusion

We have presented a text segmentation algorithm following a supervised approach which we applied to the segmentation of Greek texts. On greek text collection our algorithm outperforms Choi's and Utiyama's algorithms. This is largely important particularly in the case of texts exhibiting strong variation as far as the average length is concerned. Let us conclude this paper by discussing the reasons for this performance.

Our algorithm is characterized by (a) the use of dotplot similarity, (b) the form of our similarity function, (c) the use of a length model, (d) the use of dynamic programming, (e) the use of training data. We discuss each of these items in turn.

1. Dotplot similarity. We use a very simple similarity criterion but it is based on the dotplot and hence it captures global similarities, i.e. similarities between every pair of sentences in the document. Dotplots have also been used by Choi (Choi, 2000; Choi et al., 2001), Reynar (Reynar, 1994; Reynar & Ratnaparkhi, 1997) and Xiang and Hongyuan (Xiang & Hongyuan. 2003). On the other hand, Hearst (Hearst, 1994; Hearst & Plaunt, 1993), and Heinonen (Heinonen, 1998) use a cost function which depends only on the similarity of adjacent sentences, hence it is local. Utiyama and Isahara (Utiyama & Isahara, 2001) take an intermediate position: they use a cost function which depends on within-segment statistics, hence it is "somewhat" global, i.e. it considers similarities of all sentences within each segment. Ponte and Croft (Ponte and Croft, 1997) also use an intermediate approach, computing the similarities of all sentences which are at most n sentences apart.
2. Generalized density. We use a very simple similarity function based on a single very simple feature (i.e. we consider sentences similar when they share even a single word). However there is a special characteristic in our function, which we believe to be crucial to the success of our algorithm. Namely, we use the "generalized density" (i.e. $r \neq 2$) and this greatly improves the performance of our algorithm. Other authors have only used dotplot densities with $r = 2$ only (Choi, 2000; Choi et al., 2001; Utiyama & Isahara, 2001; Xiang & Hongyuan, 2003).
3. Length model. A term for the expected length of segments has been used by Ponte and Croft (Ponte and Croft, 1997) and Heinonen (Heinonen, 1998). Utiyama and Isahara (Utiyama & Isahara, 2001) mention the possibility but do not seem to actually use such a model. However, Choi (Choi, 2000; Choi et al., 2001), Reynar (Reynar, 1994; Reynar &

Ratnaparkhi, 1997) and several other authors do not use a length model. We have noticed that the use of the length model greatly enhances the performance of our algorithm.

4. Dynamic programming effects global optimization of the cost function and hence is a very critical factor in the success of our algorithm. As far as we know, the only other authors who have used dynamic programming are Ponte and Croft (Ponte and Croft, 1997), Heinonen (Heinonen, 1998), Xiang (Xiang & Hongyuan, 2003) and, implicitly, Utiyama and Isahara (Utiyama & Isahara, 2001) (their shortest path algorithm is actually a dynamic programming algorithm). On the other hand Choi (Choi, 2000; Choi et al., 2001) and Reynar (Reynar, 1994; Reynar & Ratnaparkhi, 1997) use divisive clustering which, strictly speaking, does not solve an optimization problem; in fact clustering performs a greedy, local optimization. Note also the heuristic approach to segmentation, first used by Hearst (Hearst, 1994; Hearst & Plaunt, 1993) and then by several other authors.
5. Training data. It should not be overlooked that our algorithm depends crucially on the availability of training data, for the estimation of the parameters μ, σ, r and γ. Training data are also used by Choi (Choi, 2000; Choi et al., 2001) for a tuning step of his clustering algorithm; Utiyama and Isahara's algorithm does not depend on training data. However, we should note that in many practical segmentation problems training data will be available (see also (Beeferman et al., 1999)).
6. Finally, for the segmentation of Greek texts we should not overlook the importance of the POS tagger; if the Greek words were not lemmatized, the vocabulary of the text collection would increase by an order of magnitude, making the segmentation problem much harder.

In short, we believe that our algorithm outperforms Choi's and Utiyama's algorithms because it performs global optimization of a global cost function. This should be compared to the local optimization of global information (used by Choi) and the global optimization of local information (used by Utiyama and Isahara).

In future work, we plan to apply our dynamic programming method to other similarity metrics such as the one proposed by Hearst (WindowDiff) in order to assess the difference in segmentation accuracy.

An interesting point would be to test our algorithm in text of continuous stream i.e. longer texts than the one used for the second experiment for the greek texts. Another interesting point to examine is to enhance the vector space model used in order to calculate the similarity between sentences with the ranking (3x3 grid which is roughly equal to the one common word measure) measure in order to avoid any stability issues that may rise by the similarity metric used by our algorithm.

In order to combine our algorithm with psychological issues such as the words used by different authors, we plan to examine some of the at least well known 1000 textual attributes relevant to authorship. The selection of those variables is based on their ability to reveal subconscious mechanisms of language variation which are unique to each author and have an impact on the discrimination of the author among every possible author, thus in our case, topic i.e. segment change. As it was proposed by Bestgen (Bestgen, 2006) our algorithm can benefit from the addition of semantic knowledge for capturing semantic relations between words appearing in sentences, which will be a future step.

6. References

Auran, C. ; Colas, A. ; Portes, C. & Vion, M. (2005). Perception of breaks and discourse boundaries in spontaneous speech: developing an on-line technique. *IDP05 - Discours et Prosodie comme Interface Complexe*.

Beeferman, D.; Berger, A. & Lafferty, J. (1999). Statistical models for text segmentation. *Machine Learning*, vol. 34, pp. 177-210.

Beeferman, D.; Berger, A. & Lafferty, J. (1997). Text segmentation using exponential models. *In Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing*, pp. 35-46.

Bertsekas, D. (1987). Dynamic Programming: Deterministic and Stochastic Models. *Prentice Hall*.

Bestgen, Y. (2006). Improving Text Segmentation Using Latent Semantic Analysis: A Reanalysis of Choi, Wiemer-Hastings Deterministic and Moore (2001). *Computational Linguistics*, vol. 1, pp. 5-12.

Choi, F.Y.Y. (2000). Advances in domain independent linear text segmentation. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 26-33.

Choi, F.Y.Y.; Wiemer-Hastings, P. & Moore, J. (2001). Latent semantic analysis for text segmentation. In *Proceedings of the 6th Conference on Empirical Methods in Natural Language Processing*, pp. 109-117.

Dias, G. & Alves, E. (2005). Unsupervised Topic Segmentation Based on Word Cooccurrence and Multi-Word Units for Text Summarization. E*LECTRA Workshop Methodologies and Evaluation of Lexical Cohesion Techniques in Real-world Applications Beyond Bag of Words (in association with SIGIR-2005)*, pp. 41-48.

Dowman, M.; Tablan, V.; Cunningham, H.; Ursu, C. & Popov, B. (2005). Semantically Enhanced Television News through Web and Video Integration. In *Proceedings of the ESWC05 Workshop on Multimedia and the Semantic Web*.

Halliday, M. & Hasan, R. (1976). Cohesion in English. *Longman*.

Hearst, M. A. (1994). Multi-paragraph segmentation of expository texts. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistic*, pp. 9-16.

Hearst, M. A. & Plaunt, C. (1993). Subtopic structuring for full-length document access. In *Proceedings of the 16th Annual International Conference on Research and Development in information Retrieval of the Association of Computer Machinery - Special Interest Group on Information Retrieval (ACM-SIGIR)*, pp. 59-68.

Heinonen, O. (1998). Optimal Multi-Paragraph Text Segmentation by Dynamic Programming. In *Proceedings of 17th International Conference on Computational Linguistics (COLING-ACL'98)*, pp. 1484-1486.

Hirschberg, J. & Litman, D. (1993). Empirical studies on the disambiguation and cue phrases. *Computational Linguistics*, vol.19, pp. 501-530.

Hsueh, P-Y.; Moore, J.D. & Renals, S. (2006). Automatic Segmentation of Multiparty Dialogue. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL) 2006*, pp. 273-280.

Kehagias, Ath.; Nicolaou A. ; Fragkou P. & Petridis V. (2004)(a). Text Segmentation by Product Partition Models and Dynamic Programming. *Mathematical and Computer Modeling*, vol. 39, pp. 209-217.

Kehagias, Ath.; Fragkou P. & Petridis V. (2004)(b). A Dynamic Programming Algorithm for Linear Text Segmentation. *Journal of Int. Information Systems*, vol. 23, pp. 179-197.

Kiat T.Y. (2005). Linear and Hierarchical Text Segmentation Using Product Partition Models. Master Thesis, Department of Computer Science, School of Computing, National University of Singapore 2004/2005.

Koshinaka, T.; Iso, K.-I. & Okumura, A. (2005). An HMM-based text segmentation method using variational Bayes approach and its application to LVCSR for broadcast news. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol.1, pp. 485- 488.

Kozima, H. (1993). Text Segmentation based on similarity between words. *In Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 286-288.

Kozima, H & Furugori, T. (1993). Similarity between words computed by spreading activation on an English dictionary. *In Proceedings of 6th Conference of the European Chapter of the Association or Computational Linguistics*, pp. 232-239.

McDonald, R.; Crammer, K. & Pereira, F. (2005). Flexible Text Segmentation with Structured Multilabel Classification. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), Association for Computational Linguistics*, pp. 987–994.

Morris, J. & Hirst, G. (1991). Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, vol. 17, pp. 21-42.

Olney, A. & Cai, Z. (2005). An Orthonormal Basis for Topic Segmentation in Tutorial Dialogue. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 2005, AAAI Press, pp. 971-978.

Orphanos, G. & Christodoulakis, D. (1999). Part-of-speech disambiguation and unknown word guessing with decision trees. In *Proceedings of EACL'99*.

Orphanos, G. & Tsalidis, C. (1999). Combining handcrafted and corpus-acquired lexical knowledge into a morphosyntactic tagger. In *Proceedings of the 2nd Research Colloquium for Computational Linguistics in United Kingdom (CLUK).*

Passoneau, R. & Litman, D.J. (1993). Intention - based segmentation: Human reliability and correlation ith linguistic cues. *In Proceedings of the 31st Meeting of the Association for Computational Linguistics*, pp. 148-155.

Pevzner, L. & Hearst, M. (2002). A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, vol.28(1), pp. 19–36.

Ponte, J. M. & Croft, W. B. (1997). Text segmentation by topic. In *Proceedings of the 1st European Conference on Research and Advanced Technology for Digital Libraries*, pp. 120 - 129.

Raskin, V., & Weiser, J. (1987). Language and Writiing: Applications of linguistics to rhetoric and composition. *Norwood, New Jersey: ABLEX: Publishing Corporation.*

Reynar, J.C. (1994). An automatic method of finding topic boundaries. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 331-333.

Reynar, J.C. & Ratnaparkhi, A. (1997). A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pp. 16-19.

Roget, P.M. (1977). *Roget's International Thesaurus*. Harper and Row, 4th edition.

Sitbon, L. & Bellot, P. (2005). Segmentation thématique par chaînes lexicales pondérées. Actes de TALN 2005, Dourdan, France.

Stamatatos, E.; Fakotakis, N. & Kokkinakis, G. (2001). Computer-based authorship attribution without lexical measures. *Computer and the Humanities*, Kluwer Academic Publishers, vol. 35, pp. 193 - 214.

Utiyama, M., & Isahara, H. (2001). A statistical model for domain - independent text segmentation. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 491-498.

Ursu, C.; Tablan, V. & Cunningham, H. (2005). Semantic Analysis for tomorrow's audio-visual digital archives. In *Proceedings of 2nd European Workshop on the Integration of Knowledge, Semantic and Digital Media Technologies (EWIMT-2005).*

Xiang J. & Hongyuan Z. (2003). Domain-independent Text Segmentation Using Anisotropic Diffusion and Dynamic Programming. In *Proceedings of the 26th ACM SIGIR Conference. on Research and Development in Information Retrieval*.

Yaari, Y. (1997). Segmentation of expository texts by hierarchical agglomerative clustering. In *Proceedings of the Conference on Recent Advances in Natural Language Processing*, pp. 59-65.

Yaari, Y. (1999). Intelligent exploration of expository texts. Ph.D. thesis. Dept. of Computer Science, Bar-Ilan University.

Ye, N.; Zhu, J.; Luo, H.; Wang, H. & Zhang, B. (2005). Improvement of the dotplotting method for linear text segmentation. *Natural Language Processing and Knowledge Engineering*, pp. 636- 641.

Appendix A: The Morphosyntactic Tagger

The Greek texts were preprocessed using the morphosyntactic tagger (better known as Part-Of-Speech tagger) developed by Ophanos et al. (Orphanos & Christodoulakis, 1999; Orphanos & Tsalidis, 1999). This is a Part-Of-Speech (POS) tagger for modern Greek (a high inflectional language) and is based on a Lexicon capable of assigning full morphosyntactic attributes (i.e. Part-Of-Speech, Number, Gender, Tense, Voice, Mood and Lemma) to 876.000 Greek word forms. Orphanos et al. created a tagged corpus capable of exhibiting the capability of the POS tagger to identify and resolve all POS ambiguity schemes present in Modern Greek (e.g. Pronoun-Clitic-Article, Pronoun-Clitic, Adjective-Adverb, Verb-Noun, etc) as well as the characteristics of unknown words by using the Lexicon. They used this corpus in order to induce decision trees, which along with the Lexicon are integrated into a robust POS tagger for Modern Greek texts. The tagger has three parts: the Tokenizer, the Lexicon and finally the Disambiguator and Guesser. The Tokenizer takes as input raw text and converts it into a stream of tokens. The Tokenizer resolves non-word tokens (e.g. punctuation marks, numbers, dates etc.) and provides them a tag corresponding to their category. As for the word tokens, they are looked up in the Lexicon and those found receive one or more tags. The Disambiguator/Guesser takes as input words that received more than one tags and words that were not found in the Lexicon and decides their contextually appropriate tag. The Disambiguator/Guesser is a 'forest' of decision trees, one tree for each ambiguity scheme present in Modern Greek and one tree for unknown guessing. The ambiguity scheme of words that received by the Lexicon more than one tag is identified and the corresponding decision tree is selected. This tree is traversed according to the values of the morphosyntactic features extracted from contextual tags. The result of this traversal is the contextually appropriate POS tag along with its corresponding lemma. In order to resolve the ambiguity, tag(s) with different POS than the one returned by the decision tree, is (are) eliminated. In order to determine the POS of an unknown word, the decision tree for unknown words is traversed and examines contextual features along with the word ending and capitalization. As a result the open class POS and the corresponding lemma of the unknown word are returned.

Appendix B

<CC>
Γ. ΔΕΡΤΙΛΗΣ ΤΟ ΒΗΜΑ, 23-03-1997 Κωδικός άρθρου: B12421B062 </CC>
<TITLE>
Σαφήνεια και αμφιβολία
</TITLE>
<TEXT>
Προϋπόθεση του καλού ύφους, η σαφήνεια είναι αναγκαία τόσο στη λογοτεχνία όσο και στην επιστημονική γραφή. Αλλά πρόκειται για δύο διαφορετικές σαφήνειες. Η μία είναι ποιητική, η άλλη εξηγηματική.
Με τη σαφήνεια του ύφους του, ο λογοτέχνης «ποιεί» την πολυσημία. Ετσι ανοίγει μπροστά στον αναγνώστη ένα ριπίδιο αναγνώσεων: τον ευκολύνει να διαβάσει και να ερμηνεύσει το πολύσημο κείμενο με πολλαπλούς τρόπους.
Αλλά ο συγγραφέας ενός επιστημονικού έργου (αυτός που κυρίως θα μας απασχολήσει σήμερα) εξαφανίζει με τη σαφήνεια του ύφους του όλες τις αμφισημίες και πολυσημίες του κειμένου. Αποκλείει έτσι τις αμφιβολίες του αναγνώστη για τα όσα ο συγγραφέας ισχυρίζεται και διευκολύνει τον ανα-γνωστικό, επιστημονικό έλεγχο. Η πολυσημία που προσπαθεί να εκφράσει ο λογοτέχνης μοιάζει, εξάλλου, αλλά δεν ταυτίζεται με την αμφιβολία που κάποτε εκφράζει στο κείμενό του ένας επιστήμονας. Την εκφράζει επειδή συναισθάνεται τα όρια του εαυτού του, του συγκεκριμένου έργου του, των προσωπικών του θεωριών, ακόμη και της επιστήμης του. Αλλά παραμένει η ανάγκη να είναι σαφείς οι θεωρίες του, σαφές και το κείμενό του. Ετσι, ο συγγραφέας από τη μια καταγράφει την αμφιβολία, από την άλλη όμως υποστηρίζει με σαφήνεια τη συλλογιστική του, τις απόψεις και τις ερμηνείες του: επειδή ο επιστημονικός λόγος, εξ ορισμού, δεν επιδέχεται αντιφάσεις.
Οπως είναι φυσικό, ο κανόνας της σαφήνειας δεν έχει ενιαία εφαρμογή. Υπάρχουν οι διαφοροποιήσεις που εξαρτώνται από την προσωπικότητα και τις ικανότητες του κάθε συγγραφέα. Ενας επιστήμονας με καλό συγγραφικό ταλέντο μπορεί ίσως να βρει ελευθεριότερους τρόπους παρουσίασης των ιδεών του, να επεκταθεί σε υπαινιγμούς, σε αμφισημίες και σε αποσιωπήσεις που έχουν τη δική τους λειτουργία και αισθητική. Αλλά αυτό δεν αναιρεί την επιστημονική του υποχρέωση να δείξει με σαφήνεια, σε άλλα σημεία του κειμένου, τις απόψεις και τις ερμηνείες του.
Υπάρχουν έπειτα διαφοροποιήσεις ανάλογες με τα γνωστικά αντικείμενα και τα είδη του γραπτού επιστημονικού λόγου. Η Ιστορία, π.χ., αφήνει περισσότερες υφολογικές δυνατότητες στον συγγραφέα από οποιαδήποτε άλλη επιστήμη. Του επιτρέπει, κάποτε του επιβάλλει κιόλας, να αναδείξει τις εσωτερικές αντιφάσεις του ανθρώπου και των ανθρωπίνων κοινωνιών· τον ρόλο των ανθρωπίνων παθών· τη σημασία των συμπτώσεων και της τύχης· το βάρος των μαζικών κοινωνικών δυνάμεων· τους

αναπόδραστους φραγμούς της φύσης.
Ωστόσο, ο ιστορικός δεν δείχνει τις αντιφάσεις ουσίας με αντιφάσεις ύφους, αλλά με σαφήνεια. Τα πάθη δεν τα δείχνει με ψευδορομαντική ασάφεια, αλλά με τη σαφήνεια εκείνη που θα αναδείξει την αιχμηρότητά τους. Τονίζει τις συμπτώσεις και την τυχαιότητα με ύφος σαφές και όχι τυχάρπαστο. Τη «μοίρα» δεν την αποδίδει σε μεταφυσικές δυνάμεις - εφόσον κάνει επιστήμη. Μπορεί να την ταυτίζει με δυνάμεις που θεωρούσαν ανεξήγητες και μεταφυσικές οι άνθρωποι που μελετά· αλλά ο ίδιος δίνει όνομα στις δυνάμεις αυτές· και τις εντάσσει, με σαφήνεια, σε έναν αιτιακό συλλογισμό, σε ένα ερμηνευτικό σχήμα.
Ενα ευτυχές ιστοριογραφικό έργο απαιτεί έναν καλό συγκερασμό της επιστήμης με την τέχνη του ύφους. Από εκεί και πέρα, υπάρχει μόνο η υπέρβαση και της επιστήμης και του ύφους. Στον υπερβατικό αυτό χώρο, εκεί όπου ο συγκερασμός γίνεται ταύτιση γνώσης και τέχνης, οδηγεί ένας δρόμος σχεδόν άβατος. Τόπος που ονειρεύονται πολλοί, επιστήμονες και τεχνίτες, τόπος άφθαστος για μας τους πολλούς - όχι, όμως, ουτοπία. Μας τον έχουν δείξει οι ελάχιστοι που έφτασαν εκεί, οι δάσκαλοί μας, ο καθένας με τη μεγάλη και τη μικρή του ιστορία, όντα διόλου μεταφυσικά, πολύ ανθρώπινα. Ενας απλός άνθρωπος δεν ήταν άραγε ο δάσκαλος που, πριν από δυόμισι αιώνες, σκάρωνε κάθε μέρα τη φυγή του προς τα εκεί, με ένα απλό, αλλά καλώς συγκερασμένο κλειδοκύμβαλο;
Ο κ. Γ. Β. Δερτιλής είναι καθηγητής της Ιστορίας στο Πανεπιστήμιο Αθηνών.
</TEXT>

Applying Artificial Intelligence to Predict the Performance of Data-dependent Applications

Paula Fritzsche, Dolores Rexachs and Emilio Luque
DACSO, University Autonoma of Barcelona
Spain

1. Introduction

Computational science (CS) is an emerging discipline that unites science and mathematics with disciplinary experience in biology, chemistry, physics, and other applied scientific fields. Within the scientific method, it is often referred to as the third science paradigm, complementing both theoretical and laboratory science (Miller & Boxer, 2005). In this work, CS involves the collaboration of the computing discipline, the mathematical support represented by the knowledge discovery process and by the models, and the computing parallel environment capability. Central to this computational science problem is the performance prediction of data-dependent applications, as shown on Figure 1. By the way, CS allows doing things that were previously too difficult to do due to the complexity of the mathematics, the large number of calculations involved, or a combination of both. Therefore, new challenges are continuously arising although CS is still at an early stage of development.

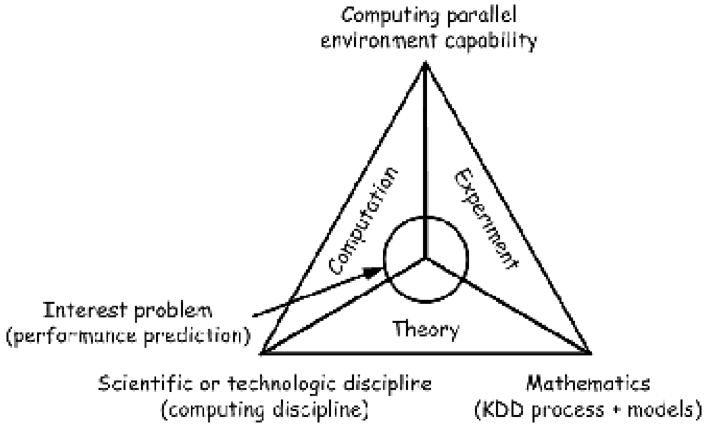

Computing parallel environment capability
Computation
Experiment
Interest problem (performance prediction)
Theory
Scientific or technologic discipline (computing discipline)
Mathematics (KDD process + models)

Figure 1. Computational science complementing both theoretical and laboratory science

Parallel computers provide an efficient and economical way to solve large-scale and/or time-constrained scientific, engineering, medicine, industry, and commerce problems. It is an alternative that makes easy to reach a solution in a fraction of the original time that would consume a single computer. Consequently, the research community, the computer designers, the professional engineers, and the end-users of these systems have a vested interest in knowing and predicting the performance order of parallel algorithms. Although measuring the performance of a parallel algorithm for all possible input values would allow answering any question about how the algorithm will respond under any set of conditions, it is impossible to make it. The situation is even worse for data-dependent algorithms where similar input data sets may cause significant variability in execution times. For this kind of algorithms, the performance does not depend only on the number of processors used (*P*) and on the data size (*N*). Other parameters have to be taken into account, the values of which are data-dependent. Great examples of this type of programs are the sorting algorithms, the searching algorithms, the satisfiability problem, the graph partition, the knapsack problem, the bin packing, the motion planning, and the traveling salesman problem (TSP). Furthermore, there are important cases of practical problems that can be formulated as TSP problems and many other problems are generalizations of this problem.

The goal of this chapter is to present a general novel methodology to the problem of predicting the performance of data-dependent algorithms. This is a good starting point for understanding some facts related with the non-deterministic algorithms. Briefly, the methodology works as follows. It begins by designing a certain number of instances and measuring their execution times. A well-designed instance guides the experimenters in choosing what experiments actually need to be performed in order to provide a representative sample. A data mining process then explores the collected data in search of patterns and /or relationships detecting the main parameters that affect performance. These common properties are modelled numerically so as to generate an analytical formulation of the execution time, a multiple-linear-regression model. Finally, the regression equation allows predicting how the algorithm will perform when given new input data sets.

A global pruning algorithm (*GP-TSP*) is used to analyze the influence of indeterminism in performance prediction, and also to show the usefulness of the proposed methodology. It is a branch-and-bound algorithm which recursively searches all possible paths and prunes large parts of the search space by maintaining a global variable containing the length of the shortest path found so far. If the length of a partial path is bigger than the current minimal length, this path is not expanded further and a part of the search space is pruned.

The *GP-TSP* execution time depends on the number of processors (*P*), number of cities (*C*), and other parameters. As a result of this investigation, right now the sum of the distances from one city to the other cities (*SD*) and the mean deviation of *SDs* values (*MDSD*) are the numerical parameters characterizing the different input data beyond the number of cities.

The preliminary experimental results of predictions are quite promising. An important fact has been reached beyond was originally sought. Choosing the city which has minimum *SD* associated value, it is possible to obtain the exact TSP solution investing less amount of time.

This chapter is organized as follows. The next section presents the novel methodology to the problem of predicting the performance of data-dependent algorithms. Section 3 reviews the traveling salesman problem (TSP) and provides detailed coverage of a parallel TSP implementation called *GP-TSP*. Section 4 focuses on the discovering process carried out to find the significant input parameters for the *GP-TSP* algorithm. Section 5 explains how to build a prediction model and then the evaluation process in order to estimate times. Finally, Section 6 summarizes and draws the main conclusions of this chapter identifying challenging future research.

2. Entire approach

The general novel methodology attempts to estimate the performance order of a parallel algorithm that solves a data-dependent problem. The defined methodology consists of three main phases: the design and composition of experiments to obtain and fit the prediction model, the validation of the model, and the use of the model developed, see Figure 2.

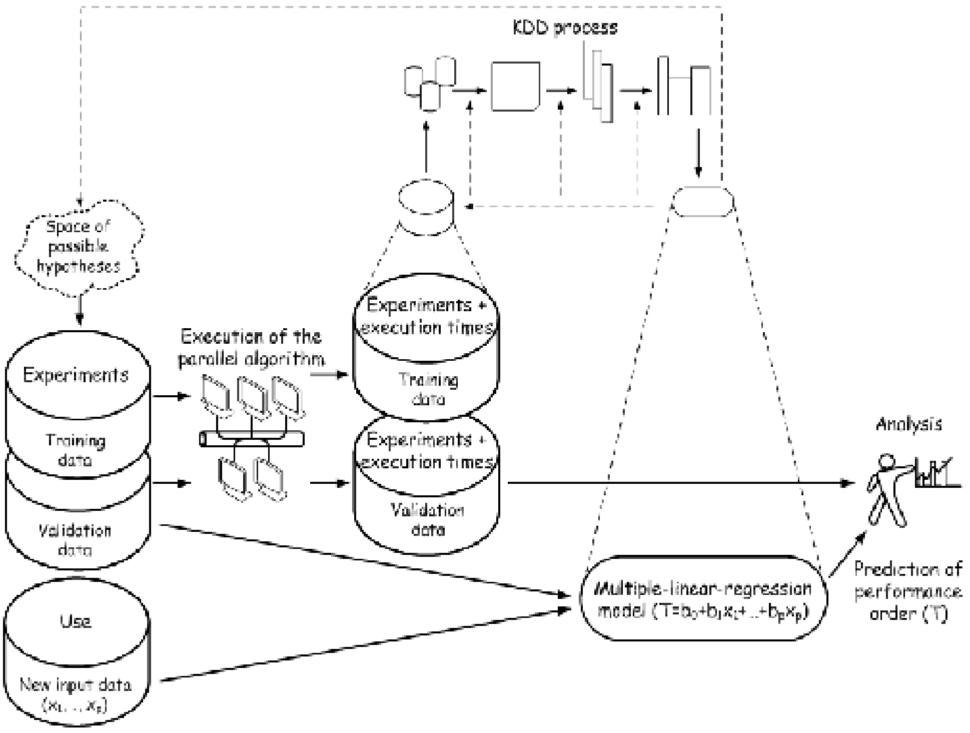

KDD process
Space of possible hypotheses
Experiments
Training data
Validation data
Use
New input data ($x_1 \ldots x_p$)
Execution of the parallel algorithm
Experiments + execution times
Training data
Experiments + execution times
Validation data
Multiple-linear-regression model ($T=b_0+b_1x_1+\ldots+b_px_p$)
Analysis
Prediction of performance order (T)

Figure 2. The performance prediction methodology

2.1 Design and composition of experiments to obtain and fit the prediction model

In principle, it is important to understand the application domain and the relevant prior knowledge, and to analyze their behavior step by step, in a deep way. It is a try-and-error method that requires specialists to manually or automatically identify the relevant parameters that can affect the execution time of the algorithm studied. Discovering the proper set of parameters is the basis to obtain a good capacity of prediction. Including too many parameters may lead to an accurate but too complicated or even unsolvable model. Hence, great care should be taken in selecting parameters and a reasonable trade-off should be made.

Designing an experiment involves articulating a goal, choosing an output that characterizes an aspect of that goal and specifying the data that will be used in the study. A well-designed instance guides the experimenters in choosing what experiments actually need to be assessed. Once a training data has been defined, the studied parallel algorithm reads and processes the experiments one by one obtaining their execution times.

A data mining application analyzes the quantitative measured values of the main parameters that affect performance and summarizes these into a useful multiple-linear-regression model (MLR model, $T=b_0+b_1x_1+\ldots+b_px_p$). It allows including the effects of several input variables that are all linearly related to a single output variable (T). This is a first approximation to deal with the problem. Figure 3 shows the knowledge construction model. Note that the instances must provide a representative sample (a training data set) first to obtain and fit the model and then to estimate the regression coefficients.

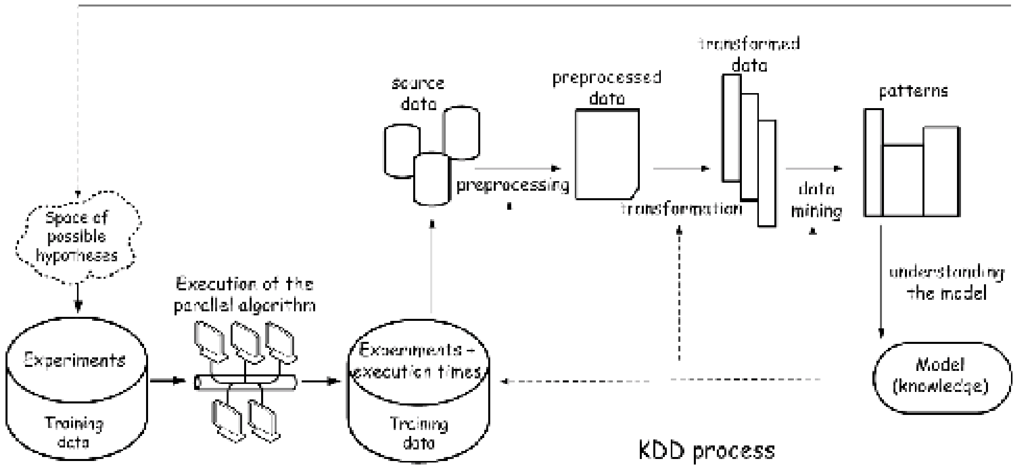


Figure 3. The knowledge construction model

2.2 Validation of the model

A new data set is used to be able to validate the created model. The validation data set constitutes a hold-out sample and is not used in building of the model. This enables to estimate the error in the predictions without having the assume that the execution times follow a particular distribution.

The training data set is used to estimate the regression coefficients ($b_0,\ldots,b_p$). These coefficients are used to make predictions for each case in the validation data. The quality analysis is a relevant issue in this stage and has to include interest measurements. The prediction for each case is then compared to the value of the dependent variable that was actually observed in the validation data obtaining the prediction error. The average of the square of this error enables to compare different models and to assess the accuracy of the model in making predictions. Figure 4 exhibits the model validation phase.

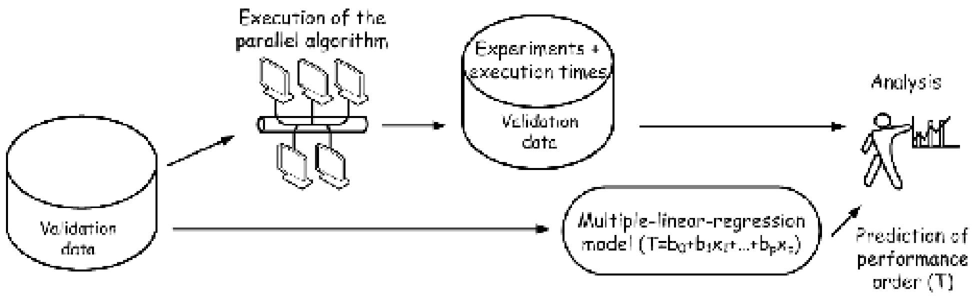

Execution of the parallel algorithm
Experiments + execution times
Validation data
Analysis
Validation data
Multiple-linear-regression model ($T=b_0+b_1x_1+\ldots+b_px_p$)
Prediction of performance order (T)

Figure 4. The validation of the model

Great care should be taken in analyzing the first approximations because it is difficult to know the degree of complexity of the relationship between the parameters and execution time. It is important to take in mind that the model aids in testing hypothesis and finding solution to performance prediction problems.
From the scientific point of view is essential to find confidence intervals for the regression parameters to provide some indication of how well they model the measured values. Taking this as a basis, it could determine the necessary number of elements in the sample.

2.3 Prediction of performance order

Once a MLR model has been fit, it is used to predict how the studied parallel algorithm will perform when given a new input data set. The $b_0,\ldots,b_p$ values are the estimated regression parameters. To predict the dependent value (T), it is necessary to replace the independent values $x_1,\ldots,x_p$ with known values.
At this point, it is necessary to emphasize that the MLR model provides a prediction framework easy to use and useful, see Figure 5.

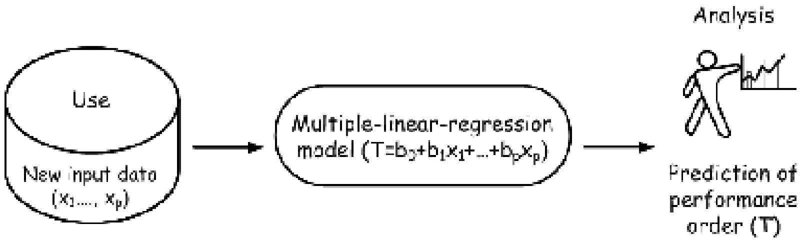

Use
New input data ($x_1,\ldots, x_p$)
Multiple-linear-regression model ($T=b_0+b_1x_1+\ldots+b_px_p$)
Analysis
Prediction of performance order (T)

Figure 5. The prediction framework

3. Traveling salesman problem

The traveling salesman problem (TSP) is one of the most famous problems (and the best one perhaps studied) in the field of combinatorial optimization. In spite of the apparent simplicity of its formulation, the TSP is a complex data-dependent. Not only the complexity of its solution has been a continue challenge to the researchers but also the prediction of its performance due to there are many practical problems that can be formulated as TSP problems.

3.1 Problem statement

The TSP for C cities is the problem of finding a tour visiting all the cities exactly once and returning to the starting city such that the sum of the distances between consecutive cities is minimized (TSP, 2008). The requirement of returning to the starting city does not change the computational complexity of the problem.

3.2 TSP computational complexity

The TSP has been shown to be NP-hard (Karp, 1972). More precisely, it is complete for the complexity class (FP^{NP})[1], and the decision problem version is NP-complete. If an efficient

[1] The class NP is the set of decision problems that can be solved by a non-deterministic Turing machine in polynomial time. FP means function problems.

algorithm is found for the TSP problem, then efficient algorithms could be found for all other problems in the NP-complete class. Although it has been shown that, theoretically, the Euclidean TSP is equally hard with respect to the general TSP (Garey et al., 1976), it is known that there exists a sub exponential time algorithm for it.

The most direct solution for a TSP problem would be to calculate the number of different tours through *C* cities. Given a starting city, it has *C-1* choices for the second city, *C-2* choices for the third city, etc. Multiplying these together it gets *(C-1)!* for one city and *C!* for the *C* cities. Another solution is to try all the permutations (ordered combinations) and see which one is cheapest. At the end, the order is also factorial of the number of cities. Generally, the presented solutions are quite similar.

3.3 TSP practical problems

Besides the drilling of printed circuits boards (Duman, 2004), transportation and logistics areas (TSP, 2008), problems having the TSP structure occur in the analysis of the structure of crystals (Bland & Shallcross, 1989), in material handling in a warehouse (Ratliff & Rosenthal, 1983), in clustering of data arrays (Lenstra & Kan, 1975), in sequencing of jobs on a single machine (Gilmore & Gomory, 1964), in physical mapping problems (Alizadeh et al., 1993), in genome rearrangement (Sankoff & Blanchette, 1997), and in phylogenetic tree construction (Korostensky & Gonnet, 2000) among others. Related variations on the TSP include the resource constrained traveling salesman problem which has applications in scheduling with an aggregate deadline (Miller & Pekny, 1991). The prize collecting TSP (Balas, 1989) and the orienteering problem (Golden et al., 1987) are special cases of the resource constrained TSP. The problem of finding a tour of maximum length is the objective in MAX TSP (Barvinok et al., 2003). The maximum scatter TSP is the problem of computing a path on a set of points in order to maximize the minimum edge length in the path. It is motivated by applications in manufacturing and medical imaging (Arkin et al., 1996). Most importantly, the TSP often comes up as a subproblem in more complex combinatorial problems, the best known and important one of which is the vehicle routing problem, that is, the problem of determining for a fleet of vehicles which customers should be served by each vehicle and in what order each vehicle should visit the customers assigned to it (Christofides, 1985).

3.4 *GP-TSP* algorithm

An implementation, called global pruning algorithm (*GP-TSP*), to obtain the exact TSP Euclidean solution in a parallel machine has been used. For simplicity of implementation, they were considered cities in R^2 instead of R^n. The most straightforward way of computing distances between cities in a two-dimensional space is to compute Euclidean distances. Anyway, the election of distance measure (Euclidean, Manhattan, Chebychev) is irrelevant. Also would be the same to work with an equivalent formulation in terms of graph theory. This is 'given a complete weighted graph (where the vertices would represent the cities, the edges would represent the roads, and the weights would be the cost or distance of that road), find a Hamiltonian circuit with the least weight' (Gutin & Punnen, 2006). Therefore, the ideas of this paper can be generalized.

The *GP-TSP* algorithm analyzes the influence of indeterminism in performance prediction. It is a branch-and-bound algorithm which recursively search all possible paths. It follows the Master-Worker programming paradigm (Fritzsche, 2007). Each city is represented by two

coordinates in the Euclidean plane. Considering C different cities, the Master defines a certain level L to divide the tasks. Tasks are the possible permutations of *C-1* cities in *L* elements. The granularity *G* of a task is the number of cities that defines the task sub-tree: *G* = *C* - *L*. At the execution start-up the Master sends the cities coordinates to every Worker.
A diagram of the possible permutations for 5 cities, considering the salesman starts and ends his trip at the city 1, can be seen in Figure 6. The Master can divide this problem into 1 task of level 0 or 4 tasks of level 1 or 12 tasks of level 2 for example. The tasks of the first level would be represented by the cities 1 and 2 for the first task, 1 and 3 for the second, followed by 1 and 4 and 1 and 5. The requirement of returning to the starting city is without detracting from the generality. In this closed cycle the salesman may begin and end in the city who wants.

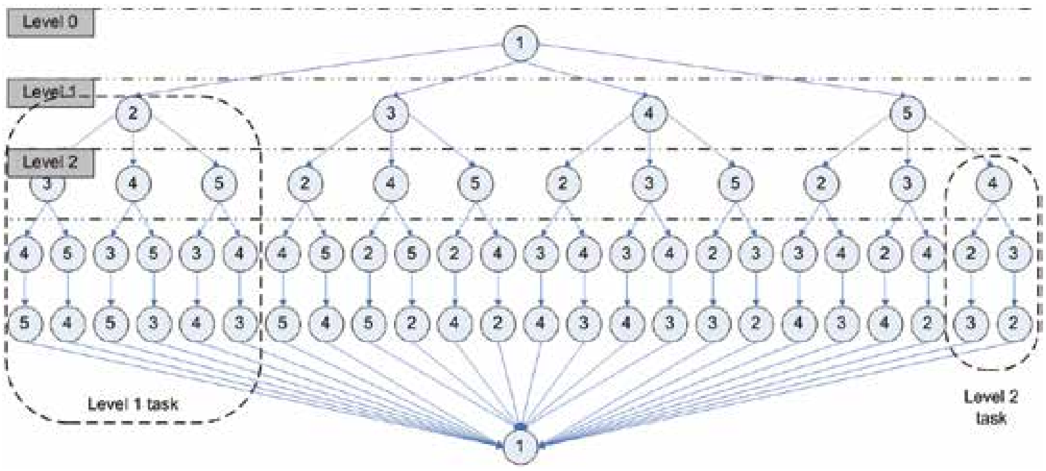
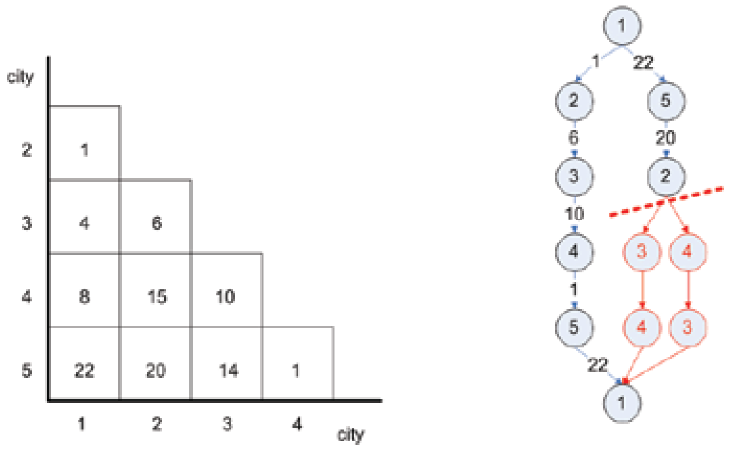


Figure 6. Possible paths for the salesman considering 5 cities

Workers are responsible for calculating the distance of the permutations left in the task and sending to the Master the best path and distance of these permutations. One of the characteristics of the TSP is that once the distance for a path is superior to the already computed minimum distance it is possible to prune this path tree.



city	1	2	3	4
2	1			
3	4	6		
4	8	15	10	
5	22	20	14	1

Figure 7. (a) Matrix of distances between cities (b) Pruning process in the *GP-TSP* algorithm

Figure 7(a) shows a strictly lower triangular matrix of distances; meanwhile Figure 7(b) exhibits the pruning process for the *GP-TSP* algorithm where each arrow has the distance between the two cities it connects. Analyzing Figure 7(b), the total distance for the first followed path (in the left) is of 40 units. The distance between 1 and 2 on the second path (in the right) is already of 42 units. It is then not necessary for the algorithm to keep calculating distances from the city 2 on because it is impossible to reach a better distance for this branch.

4. Discovering the significant *GP-TSP* input parameters

It is clear that the *GP-TSP* execution time order depends on the number of processors (*P*), on the number of cities (*C*), and 'other parameters'. Discovering the 'other parameters' is the key to obtain a good or an acceptable prediction of performance order. Undoubtedly, the knowledge discovery in databases process (KDD process) has been one of the most profitable stages in the scientific examination. A huge amount of data sets was processed with the only goal of finding some common properties. First intuitions guided the different tests in order to determine the characteristics, the relationships, and the patterns between the data sets. As a result of the investigation, right now the sum of the distances from one city to the other cities (*SD*) and the mean deviation of *SDs* values (*MDSD*) are the numerical parameters characterizing the different input data beyond the number of cities (*C*). But how these final parameters have been obtained? Next, it is described the followed way to discover the above mentioned dependencies (*SD* and *MDSD*), the construction of a model, and finally the evaluation of the obtained regression equation.

4.1 First hypothesis → location of the cities (geographical pattern)

For simplicity, only a particular training data set is analyzed and shown along different sections. It consists of five different geographical patterns of fifteen cities each one (named *G1* to *G5*). Figure 8 illustrates the five patterns handled at the beginning.

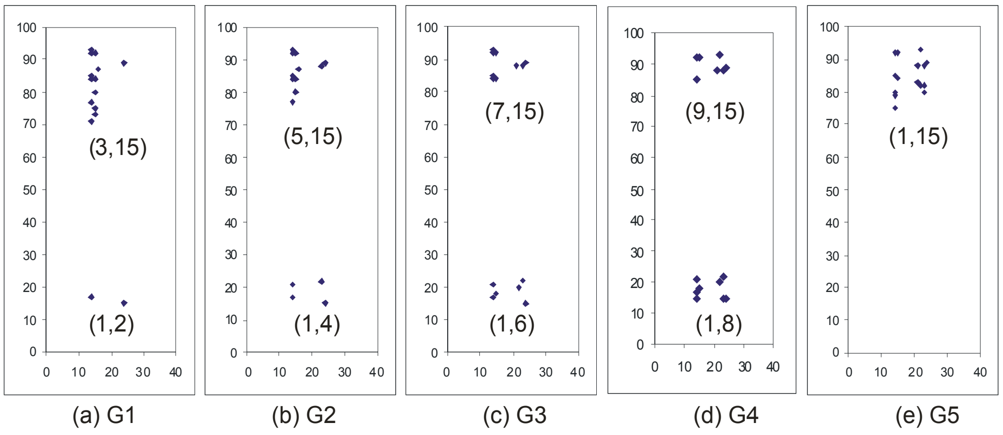


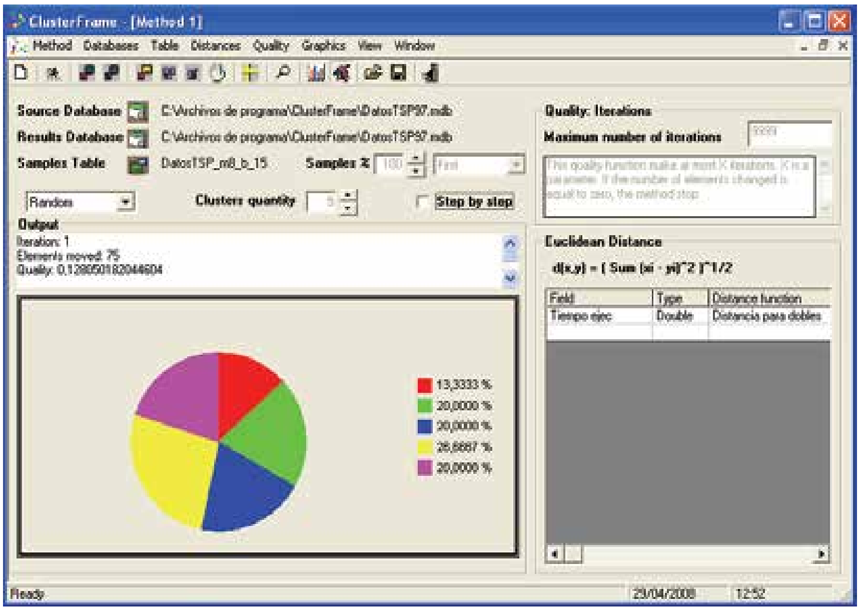
Figure 8. Five patterns defined for fifteen cities

The *GP-TSP* implementation receives the number of cities and their coordinates, and the level as input parameters. In order to find the shortest path, it proceeds recursively searching all possible paths and applying the global pruning strategy whenever it is feasible.

Hence before continuing, there are two important concepts to refresh. The main objective of data mining is finding useful patterns and knowledge in data. Clustering is one of the major data mining techniques, grouping objects together into clusters that exhibit internal cohesion (similar execution time) and external isolation.
As depicted in Figure 9, five clusters were found using a k-means algorithm (MacQueen, 1967) included in Cluster-Frame environment, see Appendix B for extra information. The idea was to obtain quite similar groups with respect to the groups (patterns) used at the beginning. The initial centroids were randomly selected by the clustering application and the squared error function, Equation (1), was the selected objective function

$$\sum_{j=1}^{k}\sum_{i=1}^{n} | x_i^{(j)} - c_j |^2 \tag{1}$$

where $|x_i^{(j)}-c_j|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centroid c_j. The entire function is an indicator of the distance of the n data points from their respective cluster centroids.



Source Database C:\Archivos de programa\ClusterFrame\DatosTSP97.mdb
Results Database C:\Archivos de programa\ClusterFrame\DatosTSP97.mdb
Samples Table DatosTSP_m8_b_15 Samples % 100 First
Random Clusters quantity 5

Output
```
Iteration: 1
Elements moved: 75
Quality: 0,128050182044604
```

Quality: Iterations
Maximum number of iterations 9999
This quality function make at most X iterations. X is a parameter. If the number of elements changed is equal to zero, the method stop

Euclidean Distance
d(x,y) = (Sum (xi - yi)^2)^1/2

Field	Type	Distance function
Tiempo ejec	Double	Distancia para dobles

Ready 29/04/2008 12:52

Figure 9. Cluster-Frame environment

Table 1 presents the obtained *GP-TSP* execution times (in sec.) by pattern (columns *G1* to *G5*) and starting city using 8 nodes of the parallel machine described in Appendix A. Columns *Cl1*,.., *Cl5* show the assigned cluster for each sample after running k-means algorithm. For the clusters 1 to 5, the final centroids values were 92.22 sec., 16.94 sec., 37.17 sec., 10.19 sec., and 7.94 sec., respectively. A simple remark derived from pattern columns is that the execution times belonging to a group are quite similar except for some cases.
The quality evaluation involves the validation of the above mentioned hypothesis. For each sample, the assigned cluster was confronted with the previously defined graphic pattern. The percentage of hits expresses the capacity of prediction. A simple observation is that the execution times were clustered in a similar way to patterns fixed at starting; the capacity of

prediction was of 75% for this example (56 hits on 75 possibilities). There was a close relationship between the patterns and the execution times.

Starting city	***Pattern***									
	G1	***Cl1***	***G2***	***Cl2***	***G3***	***Cl3***	***G4***	***Cl4***	***G5***	***Cl5***
1	216.17	1	36.50	3	15.34	2	10.51	4	8.03	5
2	214.44	1	36.82	3	15.19	2	10.49	4	7.82	5
3	77.25	1	38.09	3	15.57	2	10.02	4	7.71	5
4	72.64	1	37.29	3	15.02	2	10.30	4	7.91	5
5	70.94	1	18.54	2	15.84	2	10.41	4	7.83	5
6	74.21	1	17.83	2	15.24	2	10.24	4	7.71	5
7	75.59	1	18.16	2	10.31	4	10.36	4	7.93	5
8	73.72	1	18.03	2	10.34	4	10.26	4	7.87	5
9	69.47	1	17.79	2	10.27	4	9.98	4	8.14	5
10	74.96	1	17.48	2	10.23	4	9.88	4	8.22	5
11	75.89	1	17.07	2	10.24	4	9.85	4	8.04	5
12	70.17	1	17.39	2	10.28	4	9.87	4	8.12	5
13	73.73	1	18.10	2	10.36	4	9.88	4	7.98	5
14	70.87	1	17.37	2	10.17	4	9.95	4	8.02	5
15	73.30	1	18.00	2	10.32	4	9.97	4	7.78	5
Mean	92.23		22.97		12.32		10.14		7.94	

Table 1. Execution times (in sec.) and assigned cluster for the *GP-TSP* algorithm

Conclusions: The initial hypothesis for the *GP-TSP* was corroborated. At this stage, the asymptotic time complexity was defined as *O(C, P, pattern)*. The capacity of prediction was greater than 70% for the full range of experiments worked. This value gave evidence of the existence of other significant parameters. Therefore, a deep analysis of results revealed an open issue remained for discussion and resolution, the singular execution times by pattern. Another major hypothesis was formulated.

4.2 Second hypothesis → location of the cities and starting city

Comparing Figure 8 with Table 1 it is easy to infer some important facts. The two far cities in Figure 8(a) correspond with the two higher time values of Table 1(G1). The four far cities in Figure 8(b) correspond with the four higher execution time values of Table 1(G2). The six far cities in Figure 8(c) correspond with the six higher time values of Table 1(G3). The cities in Figure 8(d) are distributed among two zones so, the times turn out to be enough similar, see Table 1(G4). Finally, the cities in Figure 8(e) are enough closed so, the times are quite similar, see Table 1(G5).

An additional important observation is that the mean of execution times by pattern decreases as the cities approach (92.23, …, 7.94).

Conclusions: Without doubt, the location of the cities and the starting city (C_1) play an important role in execution times; the hypothesis was corroborated. At this point, the asymptotic time complexity for the *GP-TSP* was redefined as *O(C, P, pattern,* C_1*)*. Anyway, an open issue remained for discussion and resolution, how to relate a pattern (in general) with the value of the execution time. This relationship would be able to establish a numerical characterization of patterns. On this basis, a new original hypothesis was formulated.

4.3 Third hypothesis → sum of distances and mean deviation of sum of distances

What parameters could be used to quantitatively characterize different geographical patterns in the distribution of cities? Right now for each pattern, the sum of the distances from one city to the other cities (*SD*$_j$), as shown on Equation (2) and the mean deviation of *SDs* values (*MDSD*) are the worked inputs.

$$\forall j: 1 \le j \le C \quad SD_j = \sum_{i=1}^{C} \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \qquad (2)$$

In the following sub sections, three different kinds of experimentations are done. One of these is useful to see the necessity to include the both *SD* and *MDSD* parameters in the complexity expression. Another one proves that a pattern is univariate regardless of their scale or position. The last one is a singular case where the cities are uniformly distributed in a circumference.

4.3.1 Experimentation 1

Columns *SD1,.., SD5* in Table 2 show the values obtained by applying the Equation (2) to each pattern and starting city. If a particular city *j* is very remote of the others, its *SD*$_j$ will be considerably greater to the rest and consequently the execution time will grow also.

Starting city	***Pattern***									
	G1	***SD1***	***G2***	***SD2***	***G3***	***SD3***	***G4***	***SD4***	***G5***	***SD5***
1	216.17	853.94	36.50	746.10	15.34	664.60	10.51	643.75	8.03	148.74
2	214.44	887.44	36.82	740.49	15.19	649.14	10.49	635.54	7.82	104.16
3	*** 77.25**	*** 315.51**	38.09	820.63	15.57	707.70	10.02	555.70	7.71	141.15
4	**◊ 72.64**	**◊ 230.11**	37.29	789.80	15.02	678.07	10.30	599.99	7.91	103.35
5	70.94	226.88	18.54	345.83	15.84	643.65	10.41	611.45	7.83	111.79
6	74.21	244.56	17.83	330.76	15.24	638.04	10.24	595.58	7.71	102.81
7	75.59	276.09	18.16	369.56	10.31	467.99	10.36	592.68	7.93	111.28
8	73.72	294.62	18.03	383.38	10.34	490.55	10.26	639.61	7.87	147.14
9	69.47	233.53	17.79	370.10	10.27	491.52	9.98	574.23	8.14	123.19
10	**◊ 74.96**	**◊ 234.84**	*** 17.48**	*** 323.12**	10.23	446.48	9.88	578.78	8.22	172.52
11	75.89	259.19	17.07	332.87	10.24	477.42	9.85	544.61	8.04	124.64
12	70.17	234.22	17.39	325.19	10.28	449.03	9.87	534.91	8.12	131.68
13	73.73	306.99	18.10	383.11	10.36	504.79	9.88	530.72	7.98	109.78
14	70.87	239.19	17.37	327.02	10.17	451.21	9.95	574.97	8.02	124.96
15	73.30	295.27	18.00	372.00	10.32	494.09	9.97	534.36	7.78	96.29
MDSD		140.94		165.47		90.60		31.56		16.78

Table 2. Execution times (in sec.) and sum of the distances for the *GP-TSP* algorithm

Why is it needed to consider *MDSD* in addition to *SD* as a significant parameter? Quite similar *SD* values from the same experiment (same column) of Table 2 imply similar execution times. The SD_1 values for the starting cities 4 and 10 are 230.11 and 234.84, respectively. Their execution times (*G1*) are similar 72.64 and 74.96 (labelled with the symbol ◊). Instead, this relation is not true considering similar *SD* values from different patterns (different columns). The SD_1 value for starting city 3 and the SD_2 value for the starting city 10 are similar (315.51 and 323.12, respectively) but the execution times are completely dissimilar (labelled with the symbol *). Therefore, the different $MDSD(SD_1)$ and $MDSD(SD_2)$ values explain the different execution times for similar SD_1 and SD_2 values.

4.3.2 Experimentation 2

Make geometric transformations (shifting, scaling, and rotation) to well-known patterns is a fundamental test. The idea is to prove that a given pattern is univariate regardless of their scale or position. Applying each one of the transformations to a data set, similar times are expected using the same algorithm.
The coordinates of a city shifted by Δx in the *x*-dimension and Δy in the *y*-dimension are given by

$$x' = x + \Delta x \qquad y' = y + \Delta y \tag{3}$$

where *x* and *y* are the original and *x′* and *y′* are the new coordinates.
The coordinates of a city scaled by a factor S_x in the *x*-direction and *y*-direction (the city is enlarged in size when S_x is greater than 1 and reduced in size when S_x is between 0 and 1) are given by

$$x' = xS_x \qquad y' = yS_y \tag{4}$$

The coordinates of a city rotated through an angle θ about the origin of the coordinate system are given by

$$x' = x\cos\theta + y\sin\theta \qquad y' = -x\sin\theta + y\cos\theta \tag{5}$$

A data set consisting of fifteen cities is chosen from the historical database (Hist). The shifting and rotation transformations are obtained interchanging *x*-coordinate by *y*-coordinate (Sh+Rot), and the scaling transformation dividing by two both coordinates (Scaled). While Figure 10 shows these three patterns together, Table 3 exhibits a comparison of the execution times by pattern and starting city using 32 nodes of the parallel described in Appendix A. Analyzing the information by row, the historical execution times and the execution times of the geometric transformations for a sample are quite similar as it was to be expected. The mean deviations are smaller than 2%.

4.3.3 Experimentation 3

A singular case is to have the cities uniformly distributed in a circumference. The *MDSD* will be near to 0 so, similar times are expected applying any worked algorithm. Different patterns consisting of 15 to 24 cities have been studied. One of these circumferences which is composed of 24 cities is shown in Figure 11.

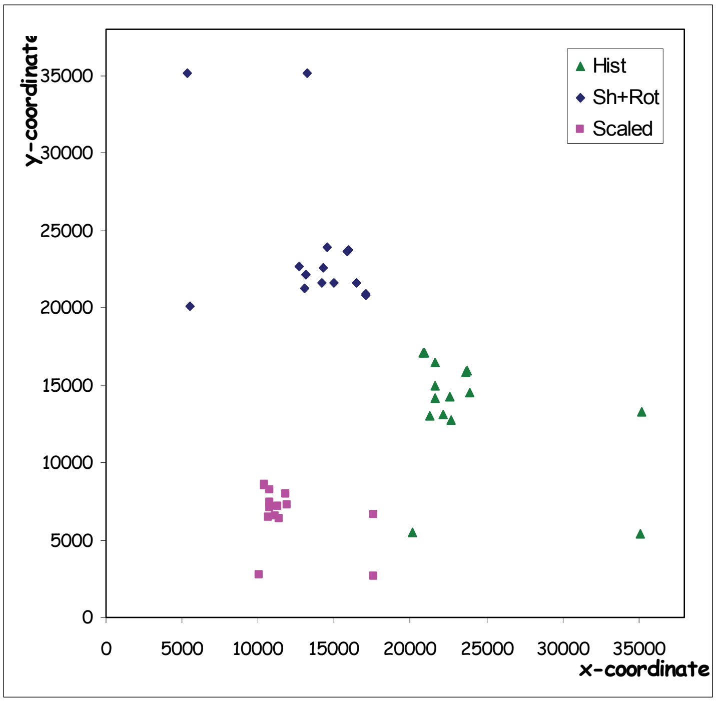


Figure 10. A historical pattern, a shifted and rotated historical pattern, and a scaled historical pattern consisting of fifteen cities

Starting city	*Pattern*			*MDev*
	Hist	*Sh+Rot*	*Scaled*	
1	46.25	48.52	47.30	0.78
2	100.30	105.60	102.77	1.81
3	73.48	76.34	74.52	1.04
4	32.92	34.52	33.75	0.54
5	30.83	31.96	31.35	0.39
6	30.49	31.92	31.22	0.48
7	31.77	33.00	32.21	0.45
8	30.10	31.06	30.43	0.35
9	31.08	32.13	31.92	0.42
10	30.98	32.24	31.60	0.42
11	29.94	31.09	30.36	0.42
12	30.33	31.53	30.85	0.42
13	31.45	32.82	32.14	0.46
14	32.67	33.44	32.53	0.37
15	32.49	33.49	32.89	0.36

Table 3. Comparison of execution times (in sec.) for the three patterns using 32 nodes

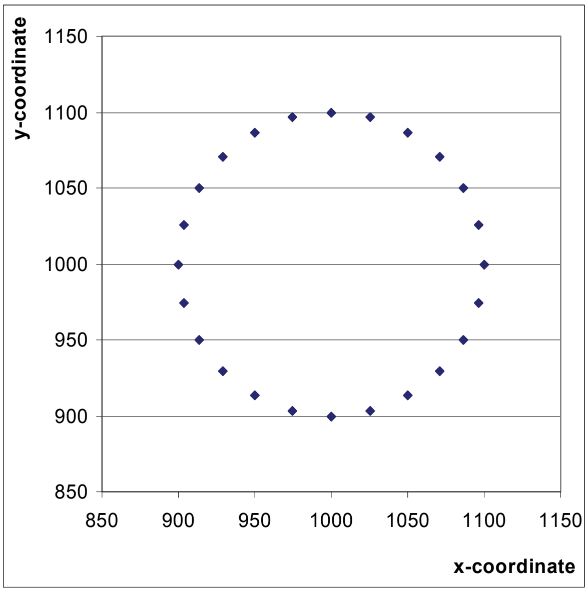


Figure 11. A circumference pattern composed of 24 uniformly distributed cities

Table 4 shows a comparative study of behaviour of different circumference patterns applying the *GP-TSP* algorithm. As it can be appreciated in Table 4, there is a minimum progressive increase in the times. It is remarkable that in every case, the mean deviations of execution times were smaller than 1%.

#Cities	***15***	***16***	***17***	***18***	***19***	***20***	***21***	***22***	***23***	***24***
Mean	12.71	17.47	23.42	32.93	42.95	54.94	68.67	129.53	367.29	1085.57
Mean deviation	0.03	0.04	0.08	0.08	0.07	0.10	0.10	0.11	0.30	2.12

Table 4. Mean and mean deviation of execution times (in sec.) by number of cities for the *GP-TSP* algorithm using 32 nodes

Conclusions: It is important to emphasize that the *GP-TSP* algorithm obtains good results of prediction. Their asymptotic time complexity should be defined as *O(C, P, SD, MDSD)*. Another important fact has been reached beyond what was originally sought. Choosing the city which has minimum *SD* associated value, it is possible to obtain the exact TSP solution investing less amount of time.

5. Predicting the *GP-TSP* execution time

The *GP-TSP* algorithm has been executed for a great amount of training patterns in order to take enough experimental data to validate this experimental approach. At this point, the methodology views the algorithm being study as a black box in which the normalized measured values for the input variables *(C, P, SD, MDSD)* arrive, are processed, and then produce a MLR model. A desired normalization converts values to a common basis for comparison. It is important to take in mind that the MLR model is a first approximation to deal with the performance prediction problem.

5.1 Building a MLR model for the *GP-TSP* algorithm

There are four independent input variables (*C, P, SD, MDSD*) and the basis form of the four-dimensional regression model for the execution time (*T*) is

$$T = b_0 + b_1 C + b_2 P + b_3 SD + b_4 MDSD \tag{6}$$

where b_0, b_1, b_2, b_3, and b_4 are the regression parameters to estimate. There exist *m* measurements of the output *T* for various combinations of the inputs *C*, *P*, *SD*, and *MDSD*. Each measurement can be expressed as

$$T_i = b_0 + b_1 C_i + b_2 P_i + b_3 SD_i + b_4 MDSD_i + e_i \tag{7}$$

where e_i is the residual for the data (C_i, P_i, SD_i, $MDSD_i$, T_i).
To find the regression parameters, it is necessary to minimize the sum of squares of the residuals, denoted *SSE*.

$$SSE = \sum_{i=1}^{m} e_i^2 = \sum_{i=1}^{m} (T_i - b_0 - b_1 C_i - b_2 P_i - b_3 SD_i - b_4 MDSD_i)^2 \tag{8}$$

The Equation (8) takes on its minimum value when the partial derivatives of *SSE* with respect to b_0, b_1, b_2, b_3, and b_4 are all set to zero. This procedure then leads to a system of five equations. The solution could be found by using any of the standard methods for solving systems of equations, or using any available software package designed for this purpose (Lilja, 2000).

5.2 Evaluating the regression equation

Finally, the regression equation is used to predict how the *GP-TSP* algorithm will perform when given new input data sets. Replacing *C*, *P*, *SD*, and *MDSM* with real values in Equation (6), it is possible to estimate the time required (*T*) to find the shortest path for this master-worker global pruning TSP algorithm.

6. Conclusions

This chapter introduces a general novel methodology to estimate the performance order of data-dependent parallel algorithms. It is important to understand that the parallel performance achieved depends on several factors, including the application, the parallel computer, the data distribution, and also the methods used for partitioning the application and mapping its components onto the architecture.
Briefly, the general methodology works as follows. It begins by designing a certain number of instances and collecting their execution-time data. A well-designed instance guides the experimenters in choosing what experiments actually need to be performed in order to provide a representative sample. A data-mining process then explores these collected data in search of patterns and/or relationships detecting the main parameters that affect performance. These common properties are modelled numerically so as to generate an analytical formulation of the execution time. The methodology views the algorithm being study as a black box in which the measured values for this limited number of inputs arrive, are processed, and then produce a multiple-linear-regression model. Finally, the regression equation allows for predicting how the algorithm will perform when given new input data sets.

A TSP parallel implementation has been studied. The *GP-TSP* algorithm analyzes the influence of indeterminism in performance prediction, and also shows the usefulness and the profits of the methodology. Their execution time depends on the number of cities (*C*), the number of processors (*P*), and other parameters. As a result of the investigation, right now the sum of the distances from one city to the other cities (*SD*) and the mean deviation of *SDs* values (*MDSD*) are the numerical parameters characterizing the different input data beyond the number of cities (*C*). The followed way to discover these proper set of parameters has been exhaustively described. Finally, their asymptotic time complexity has been defined *O(C, P, SD, MDSD)*.
Building a MLR model with the four independent input variables *(C, P, SD, MDSD)* and, then, using the regression equation, a prediction of performance order for a new data set it is possible to give. Another important fact has been reached beyond what was originally sought. Choosing the city which has minimum *SD* associated value, it is possible to obtain the exact TSP solution investing less amount of time.
This work has raised certain issues that it would be interesting to address. The utility, applicability and implementation of the methodology to other data-dependent problem still remain to be studied. Another issue concerns the problem of the obtained performance model. The existence of more or less parameters that affect performance may suggest strategies to fit the final model. Last but not least, how to provide automatic useful feedback in order to asses more studies and experiments.

Appendix

A. Specification of the parallel machine

The execution were reached with a 32 node homogeneous PC Cluster Pentium IV 3.0GHz., 1Gb DDR-DSRAM 400Mhz., Gigabit Ethernet) at the Computer Architecture and Operating Systems Department, University Autonoma of Barcelona. All the communications have been accomplished using a switched network with a mean distance between two communication end-points of two hops. The switches enable dynamic routes in order to overlap communication.

B. Specification of Cluster-Frame environment

Cluster-Frame is a dynamic and open environment of clustering (Fritzsche, 2007). It permits the evaluation of clustering methods such as K-Means, K-Prototypes, K-Modes, K-Medoid, K-Means^{+}, K-Means^{++} for the same data set. Using Cluster-Frame, the results reached applying different methods and using several parameters can be analyzed and compared.

6. References

Alizadeh, F.; Karp, R.; Newberg, L. & Weisser, D. (1993). Physical mapping of chromosomes: A combinatorial problem in molecular biology. *Symposium on Discrete Algorithms,* pp. 371-381, ACM Press.

Arkin, E.; Chiang, Y. ; Mitchell, J.; Skiena, S. & Yang, T. (1996). On the Maximum Scatter TSP, *In Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 97)*, pp. 211-220, ACM New York.

Balas, E. (1989). The Prize Collecting Traveling Salesman Problem. *Networks,* Vol.19, pp. 621-636.

Barvinok, A.; Tamir, A.; Fekete, S.; Woeginger, G; Johnson, D. & Woodroofe, R. (2003). The Geometric Maximum Traveling Salesman Problem. *Journal of the ACM,* Vol.50, No.5, pp. 641-664.

Bland, R. & Shallcross, D. (1989). Large Traveling Salesman Problems Arising from Experiments in X-ray Crystallography: a Preliminary Report on Computation. Operations Research Letters, Vol.8, pp. 125-128.

Christofides, N. (1985). Vehicle Routing. *N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, Combinatorial Optimization,* pp. 315-338, Wiley, Chichester, UK.

Duman, E. & Or, I. (2004). Precedence constrained TSP arising in printed circuit board assembly. *International Journal of Production Research,* Vol.42, No.1, pp. 67-78, 1 January 2004, Taylor and Francis Ltd.

Fritzsche, P. (2007). ¿Podemos Predecir en Algoritmos Paralelos No-Deterministas?, *PhD Thesis, University Autonoma of Barcelona, Computer Architecture and Operating Systems Department*, Spain. http://caos.uab.es/

Garey, M.; Graham, R. & Johnson, D. (1976). Some NP-complete geometric problems, STOC '76: Proceedings of the eighth annual ACM symposium on Theory of computing, pp. 10-22, Hershey, Pennsylvania, United States, ACM, New York, NY, USA.

Gilmore, P. & Gomory, R. (1964). Sequencing a One-State-Variable Machine: A Solvable Case of the Traveling Salesman Problem. Operations Research, Vol.12, No.5, pp. 655-679.

Golden, B.; Levy, L. & Vohra, R. (1987). The Orienteering Problem. *Naval Research Logistics,* Vol.34, pp. 307-318.

Gutin, G. & Punnen, P. (2006). *The Traveling Salesman Problem and Its Variations,* Springer, 0-387-44459-9, New York.

Karp, R. (1972). Reducibility among combinatorial problems: In Complexity of Computer Computations. *Plenum Press,* pp. 85-103. New York.

Korostensky, C. & Gonnet, G. (2000). Using traveling salesman problem algorithms for evolutionary tree construction. *BIOINF: Bioinformatics,* Vol.16, No.7, pp. 619-627.

Lenstra, J. & Kan, A. (1975). Some simple applications of the Travelling Salesman Problem. Operations Research Quarterly, Vol.26, No.4, pp. 717-732.

Lilja, D. (2000). *Measuring computer performance: a practitioner's guide,* Cambridge University Press, ISBN: 0-521-64105-5, New York, NY, USA.

MacQueen, J. (1967). Some Methods for Classification and Analysis of MultiVariate Observations, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol.1, pp. 281-297, L. M. Le Cam and J. Neyman, University of California Press.

Miller, D. & Pekny, J. (1991). Exact Solution of Large Asymmetric Traveling Salesman Problems. *Science,* Vol.251, pp. 754-761.

Miller, R. & Boxer, L. (2005). *Algorithms Sequential and Parallel: A Unified Approach,* Charles River Media. Computer Engineering Series, 1-58450-412-9.

Ratliff, H. & Rosenthal, A. (1983). Order-Picking in a Rectangular Warehouse: A Solvable Case for the Traveling Salesman Problem. Operations Research, Vol.31, No.3, pp. 507-521.

Sankoff, D. & Blanchette, M. (1997). The median problem for breakpoints in comparative genomics, *Proceedings of the 3rd Annual International Conference on Computing and Combinatorics (COCOON'97)*, Vol.1276, pp. 251-264, New York.

TSP page (2008). http://www.tsp.gatech.edu/history/.

Agent Systems in Software Engineering

Vasilios S. Lazarou[1], Spyridon K. Gardikiotis[2] and Nicos Malevris[2]
[1]*National & Kapodistrian University of Athens*
[2]*Athens University of Economics and Business*
Greece

1. Introduction

During the last decade the continuous growth of the Web resulted in a significant development shift from simple types of software applications to distributed multi-tier web-based applications. In general, distributed systems are by nature more complex than centralized systems. As a result, the software engineering tasks of these systems are also complicated.
Unlike traditional software applications, Web-based applications are associated with a plethora of special characteristics that impede the appliance of conventional software engineering techniques. Among them, the most important include the distributed and stateless nature of the Web, the impressively high changing frequency of implementation technologies and the spread of dynamic Web pages. Furthermore, the vital role of databases in both web and distributed applications raises a demand for introducing software engineering techniques tailored for these applications. These applications, known as database applications (DA), contain embedded SQL statements in the source code. Similarly to web applications, the presence of such special statements turns out to impose a number of limitations to the applicability of existing software engineering techniques while also originating new issues.
In this chapter, the use of agent technology to confront with the software engineering task will be illustrated. More precisely, the focus will be on the application of agent systems in order to confront with the requirements of the software engineering process for distributed software systems in general, paying particular attention to distributed database applications and web applications.
Software agents can be described as intelligent and autonomous software entities that have the ability to exhibit proactive behaviour and to collaborate with each other. The software engineering process can be greatly enhanced by utilising agent technology and adopting the architecture of an intelligent, flexible and extensible agent system. The multi-tier architecture of most distributed applications offers a suitable foundation because of its inherent complication that highlights the significant and novel contribution of a multi-agent architecture.
The rationale behind utilizing agent technology has to do with the interoperability of the software resources belonging to potentially disparate application components and disparate domains. Towards this direction, agents offer a unified platform of interaction through agent communication.
The application of agent technology for the software engineering task is certainly a new and promising research area. However, a variety of approaches that attempt to exploit the

benefits of agent technology have already made their appearance and it is expected that this tendency will further evolve. At this point, it needs to be clarified that the chapter will not focus on the research area that deals with the employment of software engineering technology for agent systems. Although similar in title, this research area deals with applying software engineering methodologies to assist the creation of multi-agent systems; something completely different.

The first one has as a goal to provide an agent infrastructure to support software testing. This is realised by suggesting multi-agent frameworks that can be used as a model to build agent systems for testing service-oriented web applications. This research track aims at presenting an agent system for tackling the issues of software maintenance and testing of distributed applications.

Illustrating the research attempts that employ software agents on software engineering tasks, they can be categorised according to two key target levels. The first one has an infrastructural target. Some research work focuses on presenting communication and coordination infrastructures for agents engaged in web software testing. Another research direction targets the creation of a multi-agent framework for software testing but the goal is on how an agent infrastructural framework can assist the job of constructing concrete agents systems for service-oriented applications.

The second one has a more applied target. As a representative work, research in which multi-agent system architectures are used in software testing of web-based applications can be mentioned. Moreover, there is ongoing research where an agent system is being utilised for the software engineering of distributed database applications. The first primary objective is to assess the maintainability and to facilitate the maintenance of such applications in the presence of changes on the schema of the underlying database. The second primary objective is to support another major software engineering task namely structural and regression software testing.

The remainder of this chapter is organised as follows. Section 2 outlines the fundamental background scientific areas of Agent Systems and Software Engineering. Section 3 introduces the first primary research direction where agent frameworks are used in software engineering. Section 4 continues the illustration covering the second primary research direction where multi-agent systems are used in software engineering. Section 5 is about Agent-Oriented Software Engineering and gives a brief description of the opposite view where the idea of an agent is being utilised as a generic software engineering model. Finally, section 6 concludes the chapter by offering an overall analysis of the current research status by highlighting the commonalities and the differences of the above research approaches, in a form of comparative evaluation, and providing a view of the scope of the current approaches and potential future research courses of action.

2. Background concepts

In this section, the background concepts relevant to the chapter are going to be illustrated. However, besides the primary concepts of Software Engineering and Agent Systems, some special topics within the research area of Software Engineering, namely Web-based Software Systems and Service-Oriented Systems, will be particularly described. The reason is that a significant amount of research that applies agent system technology to software engineering has been evolved around these topics. This section concludes by describing the current convergence of the two main concepts of this chapter.

2.1 Software engineering

Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software (IEEE, 1990). The discipline of software engineering includes knowledge, tools, and methods for software requirements, software design, software construction, software testing, and software maintenance tasks (SWEBOK, 2004). Among them, the interest is focused on the processes of software engineering that can be performed by fully automated computing techniques. In particular, such processes are software testing and software maintenance.

Software testing is the process used to assess the quality of computer software. Towards this direction, two objectives are usually identified: the verification and validation of the software. Software verification examines the way that the software is built and verifies that this matches its specifications. Software validation examines the derived software and validates that this product matches the customer requirements. In practice, software testing accomplishes its intended scope by revealing the amount of embedded software faults. Its results guide the software engineering process to reduce the amount of these faults ending up in an acceptable defect rate according to the specific software's nature. Software testing techniques are traditionally divided into *black box* and *white box* techniques. The former type treats the software as a black-box without any understanding of internal behaviour and aims to test its functionality according to its requirements. Examples of black box testing techniques include random testing, equivalence partitioning, boundary value analysis, model-based testing etc. The latter type of testing presumes that the tester has access to the source code of the software and derives tests that satisfy some code coverage or data adequacy criteria. Examples of such criteria include control flow based criteria (e.g. path, branch and statement coverage), text-based adequacy criteria (e.g. LCSAJ) and data flow criteria (e.g. definitions, uses, predicate uses, computational uses etc.).

Software maintenance is the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment (IEEE, 2004). Thus, software maintenance includes a number of both pre-delivery and post-delivery processes, which according to (IEEE, 1996) are summarized to the following: process implementation, problem and modification analysis, modification implementation, maintenance review/acceptance, migration and retirement. The maintenance processes can further be classified into categories. Among many alternative suggestions, the (ISO, 2006) proposes the four major categories of software maintenance:

- Corrective maintenance is the reactive modification of a software product performed after delivery to correct discovered problems.
- Adaptive maintenance is the modification of a software product performed after delivery to keep a software product usable in a changed or changing environment.
- Perfective maintenance is the modification of a software product after delivery to improve performance or maintainability.

Preventive maintenance is the modification of a software product after delivery to detect and correct latent faults in the software product before they become effective faults.

2.1.1 Service oriented architecture

Service oriented computing (SOC) is an emerging cross-disciplinary paradigm for distributed computing that is changing the way software applications are designed, architected, delivered and consumed (Erl, 2005). Service Oriented Architecture (SOA) is a form of distributed system architecture; its properties are consolidated by the W3C working group of web service architecture. Services are autonomous and platform-independent computational elements that can be used to build networks of collaborating applications distributed within and across organizational boundaries.

Service-Oriented Architecture (SOA) and its Web implementation Web Services (WS) promote an open standard-based and loosely coupled architecture for integrating applications in a distributed heterogeneous environment. Such applications are characterized by service orientation, task distribution, collaboration among development parties, run-time behaviour and open standards for interfacing among their components.

Service dependability is critical for establishing a trustworthy service-oriented computing environment. However, the paradigm shift from product-oriented software development to SOA and WS brings many new issues to traditional verification and validation techniques. In SOA, an application is created by dynamically discovering, binding to, and integrating the discovered services from the Internet, possibly created by third party service providers. Due to the open standards and open platform, a large number of services satisfying the same requirements can co-exist, and new services can be published at any time. Hence, during system evolution, the application can dynamically rebind to different services and the architecture can be reconfigured at runtime. The dynamic and collaborative nature of SOA brings new challenges to testing WS applications including system complexity due to the flexibility of system configuration, interoperability among third-party developed components, runtime fault detection and reliability evaluation, dynamic re-composition, and implementation transparency.

2.1.2 Web application testing

A Web application can be considered as a distributed system, with a client-server or multi-tier architecture, including the following main characteristics:

1. A wide number of users distributed all over the world and accessing it concurrently.
2. Heterogeneous execution environments composed of different hardware, network connections, operating systems, Web servers and Web browsers.
3. An extremely heterogeneous nature that depends on the large variety of software components that it usually includes. These components can be constructed of different technologies (i.e., different programming languages and models), and can be of different natures (i.e., new components generated from scratch, legacy ones, hypermedia components).
4. The ability of generating software components at run time according to user inputs and server status.

Web applications are difficult to understand and test due to lack of abstraction, highly unstructured, heterogeneous representation, mixture of presentation and application logic and dynamic page generation. Web applications testing need to address challenges introduced by new control structures like hyperlinks (navigation, request and redirection), new data flow issues (e.g., scripts that are not compiler checked, HTML/XML documents as variables, storing data as hidden elements, JSP tags-defined variables and parameters and passing data via HTTP hyperlinks) and new dynamic behaviour like navigation behaviour and Web state behaviour.

In (Di Lucca & Fasolino, 2006), they considered testing of the functional requirements with respect to four main aspects, i.e., testing scopes, test models, test strategies, and testing tools. More specifically, testing strategies define the approaches for designing test cases. They can be responsibility based (also known as black box), implementation based (or white box), or hybrid (also known as grey box). In (Nguyen, 2000) it is said that 'Gray-box testing is well suited for Web application testing because it factors in high-level design, environment, and interoperability conditions. It will reveal problems that are not as easily considered by a black-box or white-box analysis, especially problems of end-to-end information flow and distributed hardware/software system configuration and compatibility. Context-specific errors that are germane to Web systems are commonly uncovered in this process.

2.2 Agents and multi-agent systems

Agents and multi-agent systems (MAS) have recently emerged as a powerful technology to face the complexity of a variety of modern Information Systems (Zambonelli & Omicini, 2004). For instance, several industrial experiences already testify to the advantages of using agents in Web services and Web-based computational markets and distributed network management. In addition, several studies advise on the possibility of exploiting agents and MAS as enabling technologies for a variety of future scenarios, i.e., pervasive computing, grid computing and semantic web.

The core concept of agent-based computing is, of course, that of an agent. However, the definition of an agent comes along with a further set of relevant agent-specific concepts and abstractions. Generally speaking, an agent can be viewed as a software entity with the following characteristics (Jennings, 2001):

- Autonomous: an agent is not passively subject to a global, external flow of control in its actions. That is, an agent has its own internal execution activity (whether a Java thread or some other sort of goal-driven intelligent engine, this is irrelevant in this context), and it is pro-actively oriented to the achievement of a specific task on user's behalf.
- Situated: an agent performs its actions while situated in a particular environment, whether a computational (e.g., a Web site) or a physical one (e.g., a manufacturing pipeline), and it is able to sense and affect (portions of) such an environment in order to meet its design objectives.
- Social: in the majority of cases, agents work in open operational environments hosting the execution of a multiplicity of agents, possibly belonging to different stakeholders (think, e.g., of agent-mediated marketplaces). In these MAS, the global behaviour derives from the interactions among the constituent agents. In fact, agents may communicate/coordinate with each other (in a dynamic way and possibly according to high-level languages and protocols) either to achieve a common objective or because this is necessary for them to achieve their own objectives.

It is clear that an agent system cannot be simply reduced to a group of interacting agents. Instead, the complete modelling of the system requires explicitly focusing also on the environment in which the MAS and its constituent agents are situated and on the society that a group of interacting agents give rise to. Modelling the environment implies identifying its basic features, the resources that can be found in the environment, and the way via which agents can interact with it. Modelling agent societies implies identifying the overall rules that should drive the expected evolution of the MAS and the various roles that agents can play in such a society (Zambonelli et al., 2003).

2.3 Agent systems and software engineering

The emergent general understanding is that agent systems, more than an effective technology, represent indeed a novel general-purpose paradigm for software development.

Agent-based computing promotes designing and developing applications in terms of autonomous software entities (agents), situated in an environment, and that can flexibly achieve their goals by interacting with one another in terms of high-level protocols and languages.
These features are well suited to tackle the complexity of developing software in modern scenarios since:

1. The autonomy of application components reflects the intrinsically decentralised nature of modern distributed systems and can be considered as the natural extension to the notions of system modularity and encapsulation;
2. The flexible way in which agents operate and interact (both with each other and with the environment) is suited to the dynamic and unpredictable scenarios where software is expected to operate (Zambonelli et al., 2001);
3. The concept of agency provides for a unified view of artificial intelligence (AI) results and achievements, by making agents and MAS act as sound and manageable repositories of intelligent behaviours (Russel & Norvig, 2003).

2.3.1 Agent systems and web systems

Engineering distributed systems is a challenging task due to issues such as concurrency, fault tolerance, security and interoperability (Sommerville, 2004; Tsai et al., 2003). With respect to engineering web service systems, applying agent techniques to service orientation field has proven a natural choice. The research on agent-based applications has so far demonstrated that agents can glue together independently developed legacy systems. The control of a system can be distributed among autonomous agents and still maintain global coherence. Moreover, system's capability improves greatly when systems (represented by agents) cooperate.
Therefore, applying the MAS technique in WS has been a focus of WS research, such as service discovery, selection, and orchestration (Buhler & Vidal, 2003; Maamar et al., 2005; Richards et al., 2003; Sycara et al., 2001).
However, agents correspond to a broader concept with respect to services. In (Qi et al., 2005), the notion of agent-based web services (AWS) is proposed, including architecture and meta-model and integration. The key challenge is to develop an integration framework for the two paradigms, agent- and service-oriented, in a way that capitalizes on their individual strengths.

3. Agent infrastructures in software engineering

In this section, the research work relevant to defining agent infrastructural frameworks will be covered. This work targets distributed software systems in general but also web services and web-based applications in particular.

3.1 A multi-agent framework for testing distributed systems

In (Yamany et al., 2006), a design for testing distributed systems is proposed. They use a three-tier distributed system structure consisting of a server, middleware and multiple clients. The server contains the data repository of the distributed application, whereas the middleware is considered to be the software bus associated with those clients.

Agents in the proposed multi-agent architecture consist typically of two generic types: social (immobile) agents and mobile agents. Social agents are used to monitor the three-tier architecture of these distributed systems (i.e. server, middleware and clients) and to execute various scheduled testing types such as unit testing and integration testing. Moreover, mobile agents are used to carry out an urgent testing such as regression testing specified by a tester (i.e. human or an agent). In addition to that, the proposed framework monitors the user usage in order to increase the leverage of the testing process by increasing the chances to discover most of the defects that might appear in both the server and clients sides.

The framework consists of three levels of autonomous and adaptive agents. The first level of agents is on the server side. Basically, it is a single agent that monitors the data of the distributed application and is called the Database Repository Agent (DRA). The second one – Middleware Controller Agent (MCA) – is located at the middleware and is the kernel of the proposed framework. Its main goals are to investigate the middleware behaviour, collect the return feedback from the clients and make an integrated report about the system. Finally, a group of social agents is distributed over the available clients. Each one is named Client Checker Agent (CCA) and is responsible for unit testing.

The framework can be extended to execute more testing procedures at the request of the tester. In some crucial unexpected behaviour of a distributed system, the tester can ask for further testing and this can be done by sending a supportive mobile agent that could help in that mission. This agent's name is Mobile Urgent Agent (MUA).

3.2 Agent fabric for web services

In (Ma et al., 2007), MAS concepts are applied for service autonomy architecture. Service agents have three basic responsibilities. They maintain runtime operations, manage service lifecycle and control trusty communication. The first one is supported by a set of basic functions, such as service discovery, monitoring and composition. The second one is an advanced feature requiring comprehensive service modelling and governance. Agent system trustworthy is also an important issue for agent collaboration.

On the other hand, an effective communication mechanism is very important for an agent system, because autonomous systems do not stand alone without interaction with other parties. This is where fabric comes into place. Fabric in SOA context usually means a messaging environment or communication infrastructure, which makes services or applications integrated. In (Ma et al., 2007), they propose a lightweight agent fabric to serve the communications between autonomous service agents and, furthermore, cross-enterprise applications. According to the aforementioned autonomous system design requirements, XMPP (Saint-Andre, 2005) is employed as the underlying communication and message routing technology to build this kind of lightweight fabric for agents. The existing XMPP technologies are also leveraged for the trusty communication between agents.

3.3 Agent framework for web services

In (Bai et al., 2006), to address the challenges of collaborative and dynamic service-oriented testing, they present a multi-agent framework (called MAST) for testing services with agent-based technology. It is based on (Tsai et al., 2003) to facilitate web service (WS) testing in a coordinated and distributed environment. Test agents are classified into different roles which communicate through XML-based agent test protocols.

The key features of MAST are:

- Testing is decomposed into different tasks including WS specification-based test generation, centralized test planning, distributed test execution, test monitoring, and test result synthesis and analysis. Different agent types are defined to accomplish various tasks.
- Test agents are organized into groups. Each group is responsible for the execution of a test plan and is composed of a group of test runners and monitors, which are coordinated by a test coordinator.
- The mechanism is defined to dynamically generate, organize, coordinate, and monitor test agents so that testing can be adaptive to reconfiguration and re-composition of services.
- A rule-based strategy is introduced to facilitate interactively define, update, and query rules for test planning and agent coordination.

Through the monitoring and coordinating mechanism, the agents can re-adjust the test plan and their behaviour at run-time to be adaptive to the changing environment. The major testing process is decomposed into three parallel and iterative phases:

1. Test script generation to define the test cases and test scenarios;
2. Test scheduling to create and allocate the test plan to agent groups;
3. Test run to exercise the test scripts, monitor execution status, and collect results.

Service specification provides basic information of the services under test such as service interface and service flow. Rule management provides the knowledge for test scheduling. Test analysis analyzes the test data such as failure rate to evaluate the quality of services and test effectiveness. MAST supports the generic testing process and classifies the agents into seven types explained as follows:

Test Master accepts test cases from Test Generator, generates test plans and distributes them to various test groups. A set of test agents that implement a test plan are organized into a test group, which is coordinated by a Test Coordinator. Test Runners execute the test scripts, collect test results and forwards the results to Test Analyzer for quality and reliability analysis. The status of the test agents are monitored by the Test Monitor.

3.4 Agent coordination model for web services

In (Xu et al., 2006), the MAST framework (see 3.2) is utilised to propose a coordination architecture based on the reactive tuple space technique to facilitate dynamic task assignment, agent creation and destruction, agent communication, agent distribution and mobility, and the synchronization and distribution of collaborative test actions. Tuple space defines a shared memory mechanism among agents by which data are structured organized, described by tuples and retrieved by pattern matching. Adding reactivity to the tuple space means the space can have its own state and react to specific agent actions. It is a hybrid approach which combines control-driven and data-driven coordination models.

In this research, two tuple spaces are defined in MAST to manage the coordination channels and to facilitate data sharing and asynchronous coordination among test agents. Through the task tuple space, test tasks are dynamically allocated to different types of test agents according to the process defined in the scheduling. Through the result tuple space, the execution results are communicated from agents to agents. A subscription mechanism is introduced to associate programmable reactions to the events occurred and state changes on the tuple space.

3.5 An agent-based framework for testing web applications

In (Kung, 2004), an agent based framework for Web applications testing is presented. The framework is based on the BDI formalism (Rao & Georgeff, 1995) and the Unified Modelling Language (UML). The BDI architecture associates beliefs, desires and intentions with agents. Beliefs are the agents' observation about the environment and other agents. Desires are goals to be accomplished. Intentions are action plans to achieve goals. Using this framework, Web testing models and other testing objects like knowledge of the component under test (CUT) and test results are modelled as beliefs, test criteria as goals, and test activities as action plans.

The framework defines a number of abstract classes for modelling agent-oriented systems. Application specific agent types are derived from these classes to inherit model-defined features and relationships, and implement inherited abstract features. In this way, the framework enforces the BDI model but also accommodate for application specific behaviours. The abstract classes include: *Belief, Goal, Plan, Agent, Agent Communication Act*, and *Blackboard* (Kavi et al., 2003; Kung et al., 2003).

The framework also introduces a number of new diagrams: **Agent Goal Diagram (AGD)** depicts the relationships between the goals and the environment and defines the roles of agents. **Use Case Goal Diagram (UCGD)** combines the UML Use Case Diagram (UCD) and the AGD to show which use cases affect which goals and vice versa. This provides a high level guidance to Agent Sequence Diagram (ASD) construction. **Agent Domain Model (ADM)** represents the domain knowledge that is internal to an agent, including the definitions of the agent's Beliefs, Goals and Plans and their intrinsic relationships. **Agent Sequence Diagram (ASD)** depicts interactions among the beliefs, goals, plans and other objects of an agent and is a refinement of an agent. These diagrams model the behaviour of a test agent. Other diagrams introduced are the Agent Design Diagram (ADD), to document the design of an agent, and the Agent Activity Diagram (AAD) and Agent State-chart Diagram (ASCD), to model the internal activity and information flows and the internal state behaviours of agents.

3.5.1 Web application test agents

There are various types of test agents for testing the various types of Web documents. A Web application test agent is composed of the various types of a test agent. Since each type of Web document has several categories of testing methods or techniques, there are specialized agents corresponding to different categories of testing methods.

All relevant test objects are modelled as the agent's beliefs including the Web component under test (CUT), the test models representing the test objects for the CUT, the requirements or functional specification of the CUT, the test cases, and test coverage result. Goals include the test requirements or test criteria, for example, percentage of requirements coverage for black-box testing, statement coverage for white-box testing. Goals have utilities which can change due to changes of beliefs. The agent always tries to fulfil the goal with the highest utility.

The action plans of an agent are generated dynamically according to the test goal selected and the current belief of the agent. An action plan is a sequence of actions to be performed by the agent to accomplish the goal. For example, if the current statement coverage is 70%, then what are the sequences of actions that can be executed to accomplish 90% statement coverage? Since each action is associated with a cost, a rational agent should select the sequence of actions that requires the minimal cost.

Finally, the actions of a test agent are implemented by command objects, each of which implements an action and has at least the following: 1) *the activity to be performed* 2) *the costs to perform the activity* 3) *a precondition* to be satisfied and 4) *a post-condition* or effect resulting from the performance of the action.

3.6 A formal agent-based framework for testing web applications

In (Miao et al., 2007), a formal framework for testing Web applications is presented. The goal is to show how the framework assists the design of agent-based Web application testing systems. In this framework, the whole test work of the Web application can be divided into the some small test tasks or subtasks. In this work, the organization-based methodology Gaia (Zambonelli et al., 2003) for multi-agent system analysis and design is employed and extended. Gaia is a methodology for agent-oriented analysis and design. Gaia is founded on the view of a multi-agent system as a computational organization consisting of various interacting roles. For the realisation Object-Z (Smith, 2000), which is a formal specification language for modular design of complex systems, was used.
The executive part of the framework is a multi-agent system (MAS) which implements all the Web test tasks. During the analysis stage, an organization is viewed as a collection of roles. Each test task corresponds to one role. At the run time, the agent takes the role to achieve the test task or interact and cooperate with other agents to finish the test tasks. The agent can not only join or leave agent society at will, but also take or release roles at run time dynamically. The framework can be easily extended by adding new roles to provide much more functionalities for testing Web applications to further enhance the intensity of automation. At the same time, agents and roles are loosely coupled; role classes and agent classes can be designed at the same time by different teams. The internal design of the multi-agent system (MAS) is independent of the Web applications.
If a new test task arrives, and there is no corresponding role in MAS to meet it, a new role can be constructed to satisfy it. Besides, if a test task couldn't be tested enough, the corresponding role can be improved or the corresponding class of role can be re-factored. If the role does not meet the requirement, it can be deleted or replaced by a new one.
The whole framework contains four layers. At the first layer, the Test Tasks Organization defines a set of conceptual test tasks of Web applications and the relationships between test tasks. At the second layer, the Role Organization consists of a set of role classes. At the third layer, the Role Instance Space consists of role instances. Each role instance is an instance of an associated role class which was defined in role organization. At the fourth layer, the Agent Organization one consists of various agents. Agents are free to join or leave the agent organization, and they can take one or more than one role instances. An agent can not only take roles at run time, but also release them if they are not needed any more. The relationships between agents are based on the relationships between roles that are taken.

4. Multi-agent systems approaches in software engineering

In this section, the research work relevant to utilising agent systems as an approach to confront with SE tasks will be highlighted. This work targets web-based applications and distributed database applications.

4.1 An agent-based data-flow testing approach for web applications

In (Qi et al., 2006), an application of the framework introduced in (Kung, 2004) (see 3.5) is presented. In this research, a particular testing approach (Qi et al., 2005) is selected and it is

shown how the framework assists the design of agent-based web application (WA) testing systems.
The testing task can be decomposed into many small subtasks and each subtask can be completed by an autonomous agent. In particular, agent-based data-flow testing is performed at the method level, object level, and object cluster level. Each level of testing is managed by a specific type of test agent. In the process of the recommended data-flow testing, an agent-based WA testing system (WAT) will automatically generate and coordinate test agents to decompose the task of testing an entire WA into a set of subtasks that can be accomplished by test agents.
A high level test agent can create low level test agents and ask them to complete the corresponding low level testing. Based on objects shared by low level test agents, a high level test agent constructs its test models and performs the comparatively high level testing that cannot be accomplished by low level test agents. Consequently, a high level testing task is completed by the cooperation of a set of low level test agents and a high level test agent.
The testing process of the proposed approach is a hybrid of a top-down process, in which a testing task is decomposed into subtasks, and a bottom-up process, in which test agents build test models and perform data-flow testing at corresponding abstraction levels to complete the subtasks.
Similar to the data-flow testing of non-WA, data-flow testing of WA requires adequate test models and proper test criteria. A Control-flow Graph (CFG) annotated with data-flow information is a generally accepted approach to model non-WA. However, a CFG has to be extended to properly handle new features of WA.
In this design, the WAT consists of two types of test agents, a blackboard, and a test case pool. The blackboard serves as the message exchanging centre in WAT and the test case pool that stores all the test cases. The test agent (Rao & Georgeff, 1995) based on the BDI model contains beliefs (observations about the environment and other agents), desires (goals to be accomplished), and intentions (action plans to achieve goals).

4.2 An agent approach to quality assurance and testing web software

In (Zhu, 2004), the application of Lehman's theory (Lehman & Ramil, 2001) of software evolution to web-based applications is studied. It is claimed that web applications are by nature evolutionary and, hence, satisfy Lehman's laws of evolution. The essence of web applications implies that supporting their sustainable long term evolution should play the central role in developing quality assurance and testing techniques and tools. Therefore, two basic requirements of such a software environment can be identified. First, the environment should facilitate flexible integrations of tools for developing, maintaining and testing various kinds of software in a variety of formats over a long period of evolution. Second, it should enable effective communications between human beings and the environment so that the knowledge about the system and its evolution process can be recorded, retrieved and effectively used for future modification of the system.
The solution proposed in (Zhu, 2004) to meet these requirements is a cooperative multi-agent software growth environment (Zhu et al., 2000; Huo et al., 2003). In this environment, various tools are implemented as cooperative agents interacting with each other and with human users at a high level of abstraction using ontology.
The software environment consists of the two types of agents. Service agents provide various supports to the development of software systems in an evolutionary strategy. They fulfil the functional requirements of development and quality assurance and testing,

verification and validation functionalities. Management agents manage service agents and are responsible for the registration of agents' capabilities, task scheduling, and monitoring and recording agents' states and the system's behaviours. Each service agent is specialized to perform a specific functional task and deal with one representation format. They cooperate with each other to fulfil more complicated tasks.

The agent society is dynamically changing; new agents can be added into the system and old agents can be replaced by a newer version. This makes task scheduling and assignment more important and more difficult as well. Therefore, management agents are implemented as brokers to negotiate with testing service agents to assign and schedule testing activities to testing service agents. Each broker manages a registry of agents and keeps a record of their capabilities and performances. Each service agent registers its capability to a broker when joining the system. Tests tasks are also submitted to the brokers.

These agents co-exist with the application software system throughout the application system's whole lifecycle to support the modifications of the system. They monitor the evolution process and record the modifications of the system and the rationales behind the modifications. They extract, collect, store and process the information about the application system and its performance, and present such knowledge to human beings or other software tools when requested. They interact with the users and developers cooperatively.

The environment grows with the application system as new tools are integrated into the environment to support the development and maintenance of new components and as the knowledge about the system is accumulated over the time. Such a software environment is called a growth environment. It significantly differs from software development environments and run-time support environments such as middleware, where evolution is not adequately supported.

In order to enable agents to cooperate effectively with each other and with human users, they communicate with each other through a flexible and collaboration protocol and codify the contents of messages in an ontology which represents knowledge about the application domain and software engineering (Zhu & Huo, 2004). The interaction protocol is developed on the basis of speech-act.

Agent	Functionality
GWP: Get Web Page	Retrieve web pages from a web site
WPI : Web Page Information	Analyse the source code of a web page, and extract the metadata, hyperlinks and structural information from the code
WSS: Web Site Structure	Analyse the hyperlink structure of a web site, and generate a node-link-graph describing the structure
TCG: Test Case Generator	Generate test cases to test a web site according to certain testing criteria
TCE: Test Case Executor	Execute the test cases, and generate execution results
TO: Test Oracle	Verify whether the testing results match a given specification
TA: Testing Assistant	Perform as user interface and guide human testers in the process of testing
WSM: Web Site Monitor	Monitor the changes of web sites, and generate new testing tasks accordingly

Table 1. Agents for testing web applications.

4.2.1 Developing a software testing ontology

In (Zhu & Huo, 2004), the design and utilisation of a software testing ontology is proposed. This attempt has the target to enrich the approach presented in (Zhu, 2004). It represents the knowledge of software engineering and codifies the knowledge for computer processing as the contents of an agent communication language. The ontology is represented in UML at a high level of abstraction so that it can be validated by human experts. It is also codified in XML for computer processing to achieve the required flexibility and extendibility. The concepts of the ontology and the relations between them are defined while their properties are also analysed. Speech-act theory is incorporated in the system and combined with the ontology to define communication protocols and to facilitate collaborations between agents.
In order to specify this ontology, a testing concept taxonomy is introduced. Taxonomy is a way to specify and organize domain concepts. Concepts are divided related to software testing into two groups: the basic concepts and compound concepts. There are six types of basic concepts related to software testing, which include *testers*, *context*, *activities*, *methods*, *artefacts*, and *environment*. Compound concepts are those defined on the bases of basic concepts, for example, testing tasks and agent's capability. Relationships between basic concepts as well as compound concepts are also introduced. Basic relations between basic concepts form a very important part of the knowledge of software testing. Therefore, they are stored in a knowledge-base as basic facts.

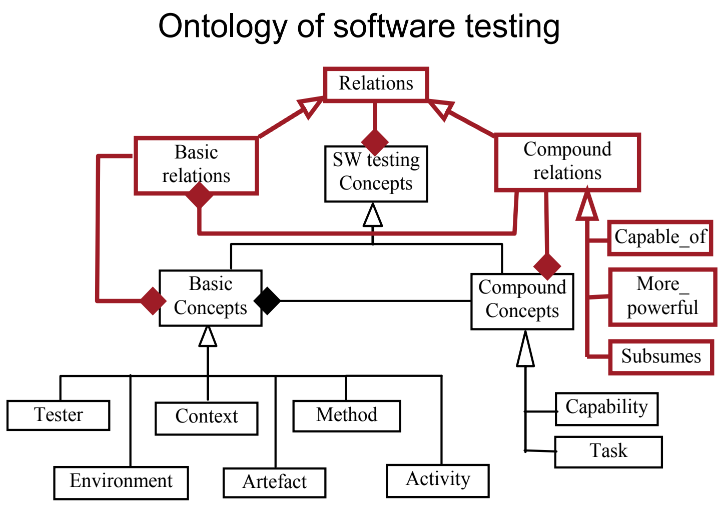


Fig. 1. Ontology of software testing

4.3 An agent approach for the maintenance and testing of database applications

In (Gardikiotis et al., 2007a), an approach for the software engineering of distributed database applications (DA) is presented. The approach is founded on the employment of software agents and adopts the architecture of an intelligent, flexible and extensible agent system that complies with the nature of multi-tier DAs. Among these agents, there are specialized agents that are capable of performing the software maintenance and testing tasks for the DAs′ source code by supporting techniques and metrics tailored for this application type. There exist also general-purpose agents that provide significant information that can be used by other DAs′ software engineering tasks (Gardikiotis et al., 2007b).

The rationale behind utilizing agent technology has to do with the interoperability of the software resources belonging to potentially disparate application components and disparate domains. Towards this direction, agents offer a unified platform of interaction through agent communication, exhibiting the following characteristics:

- Extensibility and scalability. The presented architecture can easily be extended to support other software engineering tasks. In fact, the presented system is derived from an extended version of previous work described in (Gardikiotis et al., 2007a), which focused solely on software maintenance.
- No performance degradation. The communication overhead caused by agent interaction is minimal in comparison with the process time of each individual software engineering task itself (such as the graph construction, the test case generation etc.).
- Intelligent and pro-active behaviour. The system functions in an adaptive manner by improving its mode of operation according to application complexity and coupling.
- Declarative ontology. This approach manages to encompass a customizable but formal knowledge representation to the overall agent system.

Distributed application nature. The distributed nature of the agent system fits well with the distributed nature of multi-tier applications.

4.3.1 Architecture

The architecture of the presented system is shown in Figure 2. The agents that are general in the DAs' software engineering processes are grey-coloured, whereas the maintenance agents' names are written in italics and the testing agents' names are underlined. Following a top-down approach, the role of each agent involved in the system is described.

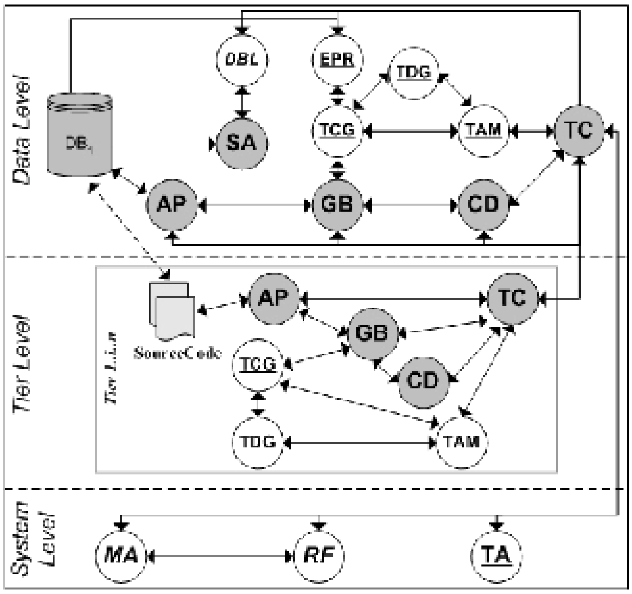


Fig. 2. Architecture

At the data level of the system, the Schema Analyzer (SA) agent stores a representation of the database schema in order to identify inter-dependencies between the database objects. The Database Listener (DBL) agent monitors the underlying databases ($DB_1 \ldots DB_n$) for any potential changes, requests from the Schema Analyzer the full set of affected database objects and can initiate the process of identifying the impact of each change into the application code by requesting this analysis from the Maintenance Assessor (MA) agent. The Execution Plan Retriever (EPR) agent retrieves from the database/s the execution plan for a specific database statement, which is given by a request from the data Tier Coordinator (TC). The TC is common for all levels, namely the data, the tier and the system level and acts as a broker, i.e. any communication between agents of different levels is transmitted through this agent. Moreover, this agent keeps track of the actual execution traces of the system that will be necessary in case of a dynamic analysis approach.

Apart from the TC, the data and the tier level share also the following agents: the Application Parser (AP), the Graph Builder (GB), the Test Cases Generator (TCG), the Test Data Generator (TDG), the Test Adequacy Measurer (TAM) and the Clustering Detector (CD). The AP parses and analyses the source code for all units included in the specific tier while the GB creates an abstract graph representation of the tier code. This representation has the form of different types of graphs that facilitate program comprehension together with the application of testing, maintenance and clustering techniques. It can also be used for the impact analysis performed by the Maintenance Assessor (MA) agent.

The graphs derived from the GB are used by the CD and the TCG. The former agent investigates the partitioning of the graph based on metrics provided by the TCG and the MA agents. The latter agent generates test cases for the provided graphs according to some adequacy criteria defined and referred to by the TAM. The produced set of test cases is given as input to the TDG which generates the corresponding set of test data.

The system level of the infrastructure includes the Maintenance Assessor (MA), the Refactorer (RF) and the Testing Assistant (TA) agents. The MA assesses the DA's maintainability with reference to the schema of the underlying database and estimates the impact of a potential change in the database schema into the application source code. It has to retrieve the units/statements that are related to the altered database objects in order to offer an indication of the workload with respect to the source code changes that might be needed to retain its operability. The RF provides specific semantic-preserving transformations that aim to increase the DA's maintainability.

Lastly, the TA triggers and controls the overall testing process. The trigger event can be either a human request or a request from the MA, which informs the TA about the effects of the maintenance process on the DAs' source code.

In this system, agents of similar functionalities may have different capabilities and they may deal with heterogeneous information formats. They can also be implemented using different algorithms and they can be executed on different platforms. Agents can enter the system and other agents can abandon the system dynamically. Therefore, agents register their capabilities to a specialized agent that the system offers, namely the matchmaker agent (MM). This agent offers a directory-like service (Lazarou & Clark, 1998) very common to the agent literature. It accepts and stores registrations and de-registrations from other agents in an internal knowledge base (KB). Task requests are also submitted to this agent in order to find other agents that provide a set of desired capabilities. After accepting such a request, the MM has the job to look up in the KB, to retrieve the agent(s) that best match the criteria and to reply to the agent that sent the request with the id(s) of the retrieved agent(s). From this point onwards, agents can employ direct communication.

With respect to ontological issues, in this work the focus is on classifying and representing software engineering concepts. A categorization widely acceptable in the software engineering community is used. This illustration (Figure 3) is based on (IEEE, 2004) and (SWEBOK, 2004). In addition some topics (e.g. testing levels), which are highly relevant to the tasks of the agents, are further analyzed.

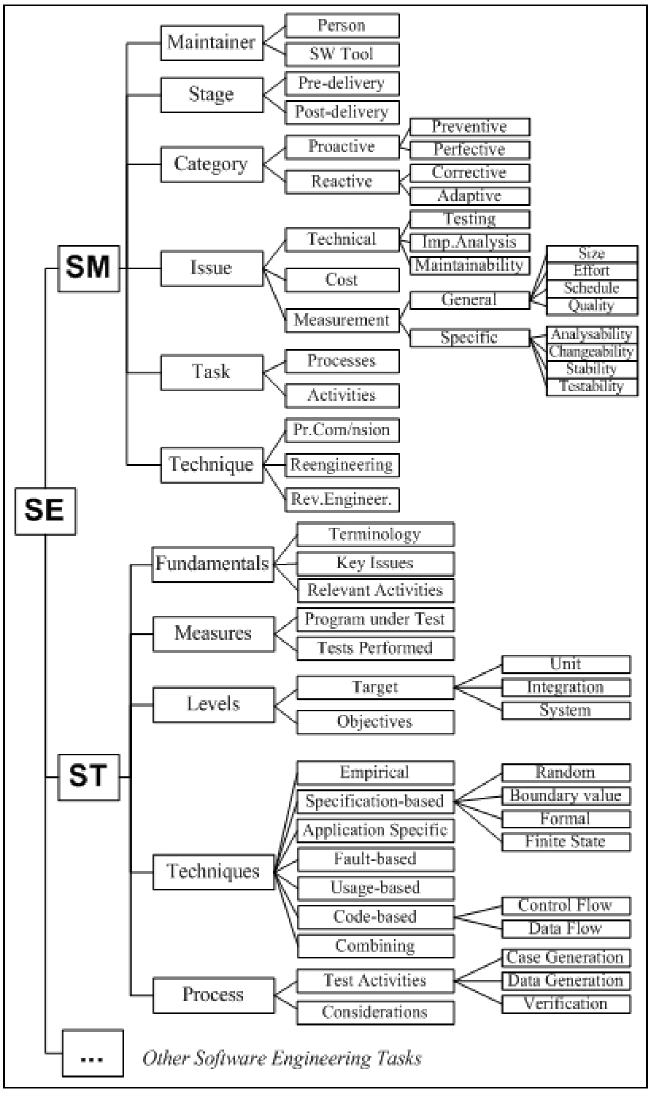


Fig. 3. Software Engineering (SE) Taxonomy

4.3.2 The agents

The agents can be categorized in three groups according to their intending tasks: software maintenance agents, software testing agents and general software engineering purpose agents.

Software Maintenance Agents

Maintenance Assessor (MA): provides an assessment of the DAs' maintainability against schema changes. It is a system-level agent that triggers and guides the maintenance process receiving a request from the data-level TC that was initially sent by the DBL. A graphical user interface is additionally provided for human user requests. To retrieve the information required for the assessment the MA communicates with the TCs. Upon the completion of its assessment task, the MA may request from the RF a set of refactorings in order to achieve a specified level of maintainability. Furthermore, it can request from the TA to trigger the testing process in order to ensure the DAs′ source code validity.

Refactorer (RF): provides a set of refactorings to increase the maintainability of the DA. Refactoring can be defined as a technique for restructuring an existing body of code, altering its internal structure without changing its external behaviour (Fowler, 1999), i.e. practically each refactoring can be viewed as a semantics preserving transformation.

Software Testing Agents

Test Case Generator (TCG): generates a set of test cases that usually refers to an abstract representation of the application source code depending on the supported technique type. The effectiveness of the generation process can be assessed by measuring the coverage of specific test adequacy criteria.

Test Data Generator (TDG): given a set of test cases the TDG automatically produces test data for them using a supported test data generation algorithm.

Test Adequacy Measurer (TAM): based on the specific testing objective the TAM proposes and measures the coverage of a set of test adequacy criteria.

Execution Plan Retriever (EPR): given a database statement the EPR retrieves from the DBMS the corresponding execution plan. This plan is necessary for the TCG to produce test cases for DAs.

Testing Assistant (TA): the TA is a system-level agent that guides the testing process. To trigger testing it either receives a request from the data-level TC or from the system-level MA agents. This request contains a description of the changes in the database schema or the DA′s source code respectively. Furthermore, the agent provides a user interface to accept requests from a human tester. The TA decides on the level of testing and the test adequacy criteria based on the available information about coupling and complexity metrics as well as the sizes and the number of DA′s clusters.

General Software Engineering Agents

Database Listener (DBL): captures the modifications made in the database schema and triggers the impact assessment.

Application Parser (AP): parses and statically analyses the DA′s unit source code. The information gained from the analysis constitutes the basis for the performance of software engineering activities such as testing and maintenance.

Graph Builder (GB): provides a set of graph representations of the DA′s source code, which is independent from the implementation language.

Tier Coordinator (TC): the TC agent serves as a local matchmaker agent (MM), i.e. it offers a directory-like service. It is aware of each tier-based agent capabilities (after receiving a corresponding register message) and uses this knowledge upon a request that is submitted by tier independent agents or other TCs located on different tiers/levels.

Schema Analyzer (SA): the SA agent resides in the data-tier and keeps a representation of the database schema in order to effectively detect dependencies between the database objects.

Clustering Detector (CD): detects the possibility of application clustering that will facilitate the software testing activities. Clustering refers to collections of source code units that are more or less relevant to activity′s target.

5. Agent-oriented software engineering

It has been already mentioned that the focus of this chapter is not about applying software engineering models to assist the creation of multi-agent systems. However, in the last years, together with the increasing acceptance of agent-based computing as a novel software engineering paradigm, there has been a great deal of research related to the identification and definition of suitable models and techniques to support the development of complex software systems in terms of MAS (Gervais et al., 2004). As a result, in order to augment the completeness of the survey, a brief depiction of this research area follows.
This research, which can be roughly grouped under the term "agent-oriented software engineering", proposes a variety of new metaphors, formal modelling approaches, development methodologies and modelling techniques, specifically suited to the agent-oriented paradigm. The current trends in this area are outlined as follows (Zambonelli & Omicini, 2004):

- Agent modelling. Novel formal and practical approaches to component modelling are required, to deal with an agent as an autonomous, pro-active, and situated entity. A variety of agent architectures are being investigated, each of which is suitable to model different types of agents or specific aspects of agents: purely reactive agents, logic agents (Van der Hoek & Wooldridge, 2003), agents based on belief, desire and intentions (Rao et al., 1995). Overall, this research has so far notably clarified the very concept of agency and its different facets.
- MAS architectures. As it is necessary to develop new ways of modelling the components of a MAS, in the same way it is necessary to develop new ways of modelling a MAS as a whole. Detaching from traditional functional-oriented perspectives, a variety of approaches are being investigated to model MAS. In particular, approaches inspired by societal, organisational, and biological metaphors, are the subject of the majority of researches and are already showing the specific suitability of the different metaphors in different application areas.
- MAS methodologies. Traditional methodologies of software development, driving engineers from analysis to design and development, must be tuned to match the abstractions of agent-oriented computing. To this end, a variety of novel methodologies to discipline and support the development process of a MAS have been defined in the past few years (Kolp et al., 2002; Wood et al., 2001), clarifying the various sets of abstractions that must come into play during MAS development and the duties and responsibilities of software engineers.
- Notation techniques. The development of specific notation techniques is needed to express the outcome of the various phases of a MAS development process; traditional object- and component-oriented notation techniques cannot easily apply. In this context, the AUML proposal (Bauer et al., 2001), extending standard UML toward agent-oriented systems, is the subject of a great deal of research and it is rapidly becoming a de facto standard.
- MAS infrastructures. To support the development and execution of MAS, novel tools and novel software infrastructures are needed. In this context, various tools are being proposed to transform standard MAS specifications (i.e., AUML specifications) into actual agent code (Bergenti & Poggi, 2002), and a variety of middleware infrastructures have been deployed to provide proper services supporting the execution of MAS.

With respect to MAS methodologies, research work involves the definition of a common framework for MAS specification, which includes the identification of a minimum set of concepts and methods that can be agreed in the different approaches (Bernon et al., 2006). The tool for defining this framework is meta-modelling. Achieving concrete results in this area would be very useful for several reasons:

1. This partly solves the lack of standardization in this area.
2. This could encourage the development of more flexible and versatile design tools.
3. This is one of the essential steps for reaching a concrete maturity in the study of the whole agent design process.

The definition of MAS meta-models has led to the identification (and formalization) of a unified meta-model. Nevertheless, the research is still in its early stages, and several challenges need to be faced before agent-oriented software engineering can deliver its promises, becoming a widely accepted and a practically usable paradigm for the development of complex software systems.

6. Conclusion

In this chapter, the application of multi-agent systems to tackle the software engineering task was outlined. The concentration was on the employment of agent technology in order to deal with distributed software systems and mainly distributed database applications and web applications.

The rationale behind utilizing agent technology has to do with the multi-tier architecture and the associated inherent complication of distributed applications and the required interoperability of software resources belonging to potentially disparate application components and disparate domains. To meet these requirements, agents offer a unified platform of interaction through agent communication.

The current research status can be classified according to two principal tracks. The first one has as a goal to provide an agent infrastructure to support software testing. This is realised by suggesting multi-agent frameworks that can be used as a model to build agent systems for testing service-oriented web applications. The second one has a more applied nature. This research track aims at presenting an agent system for tackling the issues of software maintenance and testing of distributed applications.

Analysing the aforementioned research attempts some general comments can be stated. A first and important comment is that all approaches have a quite narrow scope. On the one hand, the application domain is related to web services, web applications and database applications. The only exception is the work of (Yamany et al., 2006) but even in this case only 3-tier applications are considered. These domains have a surely specific nature even though they provide a solid basis for introducing the existing attempts.

Moreover, this restriction is made clearer by the fact that the software engineering process is not covered in its complete form. Almost all attempts target software testing with the exception of (Gardikiotis et al., 2007a) where software maintenance is also treated in depth. The above work is also the only one where the existing platform has proven its extensibility by including generic software engineering agents.

Focusing on infrastructural approaches, the work of both (Ma et al., 2007) and (Xu et al., 2006) has a very specific objective which is to support agent collaboration. Besides this commonality, the research of (Xu et al., 2006) is more tailored to software testing encompassing the notion of test tasks while the one of (Ma et al., 2007) recommends an agent design for web service autonomy. However, in both cases there is no actual system to verify the expected benefits of the two mechanisms.

With respect to agent frameworks (Yamany et al., 2006; Bai et al., 2006; Kung, 2004; Miao et al., 2007), the common aspiration is to model software testing. The testing process is decomposed into phases during test planning while these plans can be executed asynchronously. Additionally, different testing techniques can be chosen by different agents, the agent society is dynamic (agents can enter or exit the system during execution time) while the whole procedure is being coordinated by specialized agents.

The proposals of (Kung, 2004; Miao et al., 2007) offer an additional benefit that they are based on a sound formal ground employing the BDI metaphor and the Gaia methodology respectively. The work of (Kung, 2004) extends UML to put forward novel agent-oriented diagrammatic techniques that are anticipated to assist agent modelling. The research of (Miao et al., 2007) exhibits advanced flexibility since agents can change testing roles dynamically. Finally, in all cases besides (Kung, 2004) no particular approach has been bundled to validate the strength of the model's functionality while the issues of test planning optimization and agent society evolution need further exploration.

Proceeding with multi-agent systems approaches (Qi et al., 2006; Zhu, 2004; Gardikiotis et al., 2007a) they do not share many things in common. In all three approaches, agents can be designed for different tasks, deal with different representation formats and deployed on different platforms. In both (Qi et al., 2006; Zhu, 2004) the application domain is the one of web applications where test tasks are decomposed into subtasks and test agents that undertake these subtasks work together to complete the testing task. In (Qi et al., 2006) the objective is to show an implementation of (Kung, 2004) by adjusting a data-flow testing method to properly handle web applications.

The approaches of (Zhu, 2004; Gardikiotis et al., 2007a) suggest enriched architectures since they have an evolutionary and adaptive nature where existing techniques can be adapted to new application environments while new techniques can be also plugged in. Furthermore, ontological aspects are taken into consideration. Nevertheless, ontological treatment is substantially different. In (Zhu, 2004) a specialized taxonomical scheme is devised by the author to support software testing. The key offering is that besides basic concepts, compound concepts and concept relationships can be expressed. On the other hand, in (Gardikiotis et al., 2007a) the ontological representation is grounded on IEEE standards making it undoubtedly acceptable in almost any application environment. And although currently no compound concepts or concept relationships are defined, the selected representation leaves room to encompass such features in the future. In addition, a drawback of the ontological scheme proposed in (Zhu, 2004) is that it is represented in two different notations, UML and XML. This raises an issue of how to definitely ensure the consistency between them.

Concluding, the level of agent sophistication is also dissimilar. In (Zhu, 2004) agent functionalities are relatively straightforward since the focus is in other aspects. On the contrary, there are several agents that employ advanced intelligent techniques; for example the ones responsible to endorse the tasks of clustering and refactoring.

6.1 Future work

There are different future directions with respect to applying agent systems technology in software engineering. Starting with the current research status that introduces agent infrastructural frameworks, the following can be stated:

- Investigating the application of agent technology to model software engineering tasks other than software testing is obviously a desired future path.
- Applying the framework in a variety of distributed systems is absolutely necessary to optimise the model's functionality.
- Since this work has a somewhat theoretical nature, it is important that tools are developed to verify and validate the models through the use of a set of concrete test agents.
- Integrating third-party technology, methods or tools to the framework is expected to constantly increase its functionalities.
- Designing of more specific role organizations (that have to be consistent with corresponding agent organizations) and more formal definition of the mechanism of test planning is also advisable. This can include rule-based test planning, partially order plan generation and plan partitioning.

Continuing with current research relevant to agent multi-agent systems approaches, some of the remarks to be stated share some similarity to the above ones. More specifically:

- Completing the picture of the software engineering process would be a nice step forward.
- Expanding the work to handle a diversity of distributed software systems is also needed.
- The current approaches have reached a prototype level. Thoroughly testing, evaluating and deploying the agent systems, is in demand so that these approaches reach the level of a full-fledged ready to use system.
- Implementing an even richer variety of test agents. Especially, it would be really significant to employ deeper intelligent techniques (coming from the Machine Learning literature for example) in order to enhance the agent capabilities.
- Establishing a common ontological representation. This representation has the goal to be on the one hand readable and declarative from the human point of view and on the other hand flexible and able to be captured from the machine part. An agent-oriented modelling language such as AUML could prove necessary to catch the agents' autonomous and social behaviours.

A more detailed comment about web systems is that extending the current work to handle dynamically generated Web pages and to incorporate automatic test case generation techniques such as navigation testing and object state testing would refine the agent approach.

7. References

Bai, X.; Dai, G.; Xu, D. & Tsai, W.T. (2006). "A Multi-Agent Based Framework for Collaborative Testing on Web Services", Proceedings of the 2nd International Workshop on Collaborative Computing, Integration, and Assurance, WCCIA 2006, Page(s): 205-210.

Bauer, B.; Muller, J. P. & Odell, J. (2001). ''Agent UML: A formalism for specifying multi-agent software systems,'' Int. J. Soft. Eng. Knowl. Eng. vol. 11, no. 3, pp. 207 – 230.

Bergenti, F. & Poggi, A. (2002). ''Agent-oriented software construction with UML,'' in The Handbook of Software Engineering and Knowledge Engineering - volume 2 - Emerging Technologies, World Scientific: Singapore, pp. 757 – 769.

Bernon, C.; Cossentino, M. & Pavon, J. (2005). Agent-oriented software engineering. *The Knowledge Engineering Review*, Vol. 20, no. 2, pp. 99-116, June 2005

Buhler, P. & Vidal, J. M. (2003). "Semantic Web Services as Agent Behaviours," in Agent-cities: Challenges in Open Agent Environments, LNCS/LNAI, B. Burg, J. Dale, et al., Eds. Berlin: Springer-Verlag.

Di Lucca, G.A. & Fasolino, A.R. (2006). Testing Web-based applications: The state of the art and future trends. Information and Software Technology. 48, 12 (Dec. 2006), 1172-1186.

Erl, T. (2005). Service-oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, NY, US.

Fowler, M. (1999). Refactoring: Improving the Design of Existing Code. Addison-Wesley, 0201485672, 1999.

Gardikiotis, S. K.; Lazarou, V. S. & Malevris, N. (2007a). "An Agent-based Approach for the Maintenance of Database Applications", *Proc. 5th Int. Conference on Software Engineering Research and Applications*, IEEE Computer Society, pp.558-565, Busan, Korea, August 2007.

Gardikiotis, S. K.; Lazarou, V. S. & Malevris, N. (2007b). "Employing Agents towards Database Applications Testing", *Proc.* 21st International Conference on Tools with Artificial Intelligence (ICTAI'07), IEEE Computer Society, pp. 157-166, Patras, Greece, October 2007.

Gervais, M.; Gomez, J. & Weiss, G. (2004). ''A survey on agent-oriented software engineering researches,'' in: Methodologies and Software Engineering for Agent Systems, Kluwer: New York (NY).

Huo, Q.; Zhu, H. & Greenwood, S. (2003). A Multi-Agent Software Environment for Testing Web-based Applications, Proc. of COMPSAC'03, Dallas, 2003, 210-215.

IEEE (1990). IEEE Standard Glossary of Software Engineering Terminology, 610.12-1990

IEEE (2004). IEEE Standard for Software Maintenance, IEEE, 2004 {IEEE1219-2004}.

IEEE (1996). IEEE/EIA 12207.0-1996//ISO/IEC12207:1995, Industry Implementation of Int. Std.ISO/IEC 12207:95, Standard for Information Technology - Software Life Cycle Processes, IEEE, 1996 {IEEE12207.0-96}.

ISO (2006). ISO/IEC Standard for Software Engineering - Software Life Cycle Processes – Maintenance, ISO/IEC, 2006 {ISO/IEC 14764}.

Jennings, N. R. (2001). ''An agent-based approach for building complex software systems,'' Commun. ACM, vol. 44, no. 4, pp. 35 – 41.

Kavi, K.; Kung, D.; Bhambhani, H.; Pancholi, G. Kanikarla, M. & Shah, R. (2003). "Extending UML to Modelling and Design of Multi-Agent Systems," In Proc. of ICSE 2003 Workshop on Software Engineering for Large Multi-Agent Systems (SELMAS), Portland, Oregon, May 3–4, 2003.

Kolp, M.; Giorgini, P. & Mylopoulos, J. (2002). ''A goal-based organizational perspective on multi-agent architectures,'' in Intelligent Agents VIII: Agent Theories, Architectures, and Languages, vol. 2333 of LNAI, Springer-Verlag, pp. 128 – 140.

Kung, D.; Bhambhani, H.; Nwokoro, S.; Okasha, W.; Kambalakatta, R. & Sankuratri, P. (2003). "Lessons learned from software engineering multi-agent systems," Proc. of IEEE COMPSAC'03, Dallas, Texas, November 3–6, 2003.

Kung, D. (2004). "An agent-based framework for testing Web applications", Proceedings of the 28th Annual International Computer Software and Applications Conference, COMPSAC 2004, vol.2, Page(s): 174-177.

Lazarou, V. S. & Clark, K. L. (1998). "Agents for Hypermedia Information Discovery", Agents' World 98, *Co-operative Information Agents, Springer-Verlag Lecture Notes in Artificial Intelligence* (1435), 1998.

Lehman, M. M. & Ramil, J. F. (2001). Rules and Tools for Software Evolution Planning and Management. Annals of Software Engineering, Special Issue on Software Management, 11(1), 15-44.

Ma, Y.F.; Li, H.X. & Sun, P. (2007). A Lightweight Agent Fabric for Service Autonomy. Proc. of International Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering, 2007, LNCS 4504, pp. 63-77, Springer-Verlag

Maamar, Z.; Kouadri-Most´efaoui, S. & Yahyaoui, H. (2005). "Towards an Agent-based and Context-oriented Approach for Web Services Composition," IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 5, pp. 686-697.

Miao, H.; Chen, S. & Qian, Z. (2007). A Formal Open Framework Based on Agent for Testing Web Applications, International Conference on Computational Intelligence and Security, 2007, pp. 281-285, 0-7695-3072-9

Nguyen, H.Q. (2000). Testing Applications on the Web: Test Planning for Internet-Based Systems, John Wiley & Sons, Inc.

Qi, Y.; Kung, D. & Wong, E. (2005). "An Agent-Based Testing Approach for Web Applications", Proceedings of Computer Software and Applications Conference, COMPSAC 2005, Volume 2, Page(s): 45-50.

Qi, Y.; Kung, D. & Wong, E. (2006). "An agent-based data-flow testing approach for Web applications." Information and Software Technology. 48, 12 (Dec. 2006), 1159-1171.

Rao, A.S. & Georgeff, M. (1995). BDI agents: from theory to practice, in: Proceedings of the First International Conference on Multi-Agent System (ICMAS'95), San Francisco, CA, USA, pp. 312–319.

Richards, D.; van Splunter, S.; Brazier, E. & Sabou, M. (2003). "Composing web services using an agent factory," In Prc. of the 1st Workshop Web Services arid Agent Based Engineering, Sydney, Australia.

Russel, S. & Norvig, P. (2003). Artificial Intelligence: A Modern Approach, Prentice Hall/Pearson Education International: Englewood Cliffs (NJ), (2nd Edn), 2003.

Saint-Andre, P. (2005). Streaming XML with Jabber/XMPP, IEEE Internet Computing, Volume 9, Issue 5, Page(s):82 – 89.

Smith, G. (2000). The Object-Z Specification Language. Advances in Formal Methods. Kluwer Academic Publishers.

Sommerville, I. (2004). Software Engineering, 7th edition, Addison-Wesley, 2004.

SWEBOK (2004). Guide to the Software Engineering Body of Knowledge (February 6, 2004). Retrieved on 2008-02-21.

Sycara, K.; Paolucci, M.; Soundry, J. & Srinivasan, N. (2001). "Dynamic discovery and coordination of agent-based semantic web services," IEEE Computing, 8(3):66-73.

Tsai, W.T.; Paul, R.; Yu, L.; Saimi, A. & Cao, Z. (2003). "Scenario-Based Web Service Testing with Distributed Agents," IEICE Transaction on Information and System, Vol. E86-D, No. 10, pp. 2130-2144.

Van der Hoek, W. & Wooldridge, M. (2003). ''Towards a logic of rational agency,'' Logic J. IGPL, vol. 11, no. 2, pp. 135 – 160.

Wood, M.; DeLoach, S. A.; & Sparkman, C. (2001). ''Multi-agent system engineering'', Int. J. Software Eng. Knowl. Eng., vol. 11, no. 3, pp. 231 – 258.

Xu, D.; Bai, X.; & Dai, G. (2006). "A Tuple-Space-Based Coordination Architecture for Test Agents in the MAST Framework," Second IEEE International Symposium on Service- Oriented System Engineering (SOSE'06), pp. 57-66, 2006

El Yamany, H.F., Capretz, M.A.M. & Capretz, L.F. (2006) A Multi-Agent Framework for Testing Distributed Systems, IEEE COMPSAC 3rd International Workshop on Quality Assurance and Testing Web-Based Applications (QATWBA'2006), Chicago, IL, USA, September, pages 151–156.

Zambonelli, F.; Jennings, N.; Omicini, A.; & Wooldridge, M. (2001). ''Agent-oriented software engineering for internet applications,'' in Coordination of Internet Agents: Models, Technologies, and Applications, Springer-Verlag: Berlin (D), pp. 326 – 346.

Zambonelli, F.; Jennings, N.; & Wooldridge, M. (2003). ''Developing multi-agent systems: The Gaia methodology,'' ACM Trans. Soft. Eng. Meth., vol. 12, no. 3, pp.417 – 470.

Zambonelli, F. & Omicini, A. (2004). Challenges and Research Directions in Agent-Oriented Software Engineering, *Journal of Autonomous Agents and Multi-agent Systems* 9(3), pp. 253-284.

Zhu, H., Greenwood, S., Huo, Q. & Zhang, Y. (2000). Towards agent-oriented quality management of information systems, Workshop Notes of 2nd International Workshop on Agent-Oriented Information Systems at AAAI'2000, Austin, USA, July 30, 2000, 57-64.

Zhu, H. & Huo, Q. (2004). Developing A Software Testing Ontology in UML for A Software Growth Environment of Web-Based Applications, Software Evolution with UML and XML, Hongji Yang (eds.), Idea Group Inc, 2004.

Zhu, H. (2004). "Cooperative agent approach to quality assurance and testing Web software", Proceedings of COMPSAC 2004 vol.2, Page(s): 110-113.

A Joint Probability Data Association Filter Algorithm for Multiple Robot Tracking Problems

Aliakbar Gorji Daronkolaei, Vahid Nazari, Mohammad Bagher Menhaj, and Saeed Shiry
Amirkabir University of Technology, Tehran, Iran

1. Introduction

Estimating the position of a mobile robot in a real environment is taken into account as one of the most challenging topics in the recent literature (Fox et al., 2000). This problem can be usually explored in two ways. Firstly, a mobile robot should be able to have knowledge about its current position. The Dead-reckoning of a mobile robot may be used to update the position of the robot assuming the initial position is known. However, the encoders of a robot cannot provide precise measurements and, therefore, the position obtained by this way is not reliable. To achieve more accurate approximation of a robot's position, measurements obtained by sensors set on a robot are used to correct the information provided by the encoders. If the mapping of a physical environment is known, the above-mentioned procedure can be easily accomplished by using some well-known approaches such as Kalman filtering (Kalman & Bucy, 1961) to localize the exact position of a mobile robot (Siegwart & Nourbakhsh, 2004). However, when there is not any knowledge about the map, mapping and localization should be conducted simultaneously. The aforementioned topic is known as Simultaneous Localization and Mapping (SLAM) in the literature (Howard, 2005).
In many applications, one may intend to localize other robots' position via a reference robot. Robot soccer problems or people tracking scenarios can be fallen in the pre-mentioned category. Although this problem appears similar to the common localization algorithms, the traditional approaches can not be used because the reference robot does not access to the odometry data of each mobile robot used in localization algorithms to predict the future position of the robot. This issue may be completely perceivable in the people tracking scenario because there is not any information about the movement of people. In this case, some models should be proposed to represent the movement of each object. By defining a suitable motion model for each target and using measurements provided by a reference robot about the current position of the moving object, a linear/nonlinear state space model is constructed representing the movement of each object.
The above-discussed topic can be fallen in the category of target tracking problems where the final aim is defined as tracking the position of a mobile object by a reference sensor. Because of inaccurate data obtained by sensors and uncertain motion models which may not provide reliable prediction of an object's movement, filtering algorithms are used to extract

the position of a mobile target. Kalman filtering has been the first method applied to the field of target tracking. However, the Kalman method and, even, its generalized form known as extended Kalman filter (EKF) (Anderson & Moore, 1979) do not provide reliable results for nonlinear state space models. This problem is very common in tracking applications where the sensor algebraic equations are usually nonlinear towards the position of a target. To remedy the above problems, nonlinear filtering using the particle filter algorithm has been proposed (Gordon et al., 1993), (Doucet et al., 2001). Particle filtering has been extensively applied to many real themes such as aircraft tracking (Ristic et al., 2004), target detection/tracking (Ng et al., 2004), navigation (Gustafsson et al., 2002), training artificial neural networks (Freitas 1999), control (Andrieu et al., 2004), etc. Besides the ease of use, particle filter algorithms lead to much more accurate results than kalman based approaches. Recently, the combination of particle and Kalman filtering has been also applied to many tracking applications, specially, when some parameters of a motion model may be also estimated beside the position of a target (Sarkka et al., 2005).
Multiple robot tracking is another attractive issue in the field of mobile robotics. This area can be imagined as a generalized type of the common tracking problem. In other words, a reference robot should localize other robots/agents position based on information obtained by sensors. To do so, measurements should be associated to the appropriate target. Moreover, some measurements may have been received from unwanted targets usually known as clutters. The combination of the data association concept and common filtering approaches has been used in the literature as the joint probability data association filter (JPDAF) algorithm (Vermaak et al., 2005). This algorithm has been greatly used in many applications such as multiple target tracking (Li et al., 2007), (Fortman et al., 1983), people tracking (Schulz et al., 2003), and security planning (Oh et al., 2004). However, in the field of multiple robot tracking, no comprehensive work has been done and many problems are yet open. For example, unlike the traditional multiple target tracking scenarios in which sensors may be assumed to be fixed or conduct an independent movement, a reference robot can make an organized movement to track other robots position much more precisely. In other words, the motion of a reference robot must be planned so that the robot can track other robots much better. Although this case has been discussed in the literature as observer trajectory planning (OTP) (Singh et al., 2007), proposed approaches are usually implemented in an offline mode. This problem may not be so desirable in multiple robot tracking scenarios where a reference robot should localize other robots simultaneously.
In this paper, some improvements are made on the traditional JPDAF algorithm for multiple robot tracking applications. To provide a better representation of a robot's movement, different motion models proposed in the literature are used to evaluate the efficiency of tracking. Moreover, a new fuzzy controller is proposed to find an optimal trajectory for the movement of the reference robot. It will be shown that this fuzzy controller minimizes the sum of distances between the reference robot and other mobile objects.
To maintain all of above-mentioned topics, this paper is organized as follows. Section 1 deals with the general theory of the JPDAF algorithm. The particle filter algorithm and the concept of data association will be covered in this section. Section 3 discusses the JPDAF algorithm for multiple robot tracking. In this section of the paper, different motion models describing the movement of a mobile robot are represented. Section 4 is devoted to present fuzzy logic controller for optimal observer trajectory planning. This section proposes a fuzzy controller which can join with the JPDAF algorithm to enhance the quality of tracking.

Simulation results confirming the superiority of our proposed algorithm are provided in section 6. Finally, section 7 concludes the paper.

2. The JPDAF algorithm for multiple target tracking

In this section, the JPDAF algorithm considered as the most widely and successful strategy for multi-target tracking under data association uncertainty is presented. The Monte Carlo version of the JPDAF algorithm uses the common particle filter approach to estimate the posterior density function of states given measurements $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$. Now, consider the problem of tracking N objects. $\mathbf{x}_t^k$ denotes the state of these objects at time t where k=1,2,..,N is the target number. Furthermore, the movement of each target can be described in a general nonlinear state space model as follows:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t) + \mathbf{v}_t \tag{1-1}$$

$$\mathbf{y}_t = \mathbf{g}(\mathbf{x}_t) + \mathbf{w}_t \tag{1-2}$$

where $\mathbf{v}_t$ and $\mathbf{w}_t$ are white noises with the covariance matrixes Q and R, respectively. Also, in the above equation, $\mathbf{f}$ and $\mathbf{g}$ are the general nonlinear functions representing the dynamical behavior of the target and the sensor model. The aim of the JPDAF algorithm is to update the marginal filtering distribution for each target $p(\mathbf{x}_t^k \mid \mathbf{y}_{1:t})$, k=1,2,..,N, instead of computing the joint filtering distribution $p(\mathbf{X}_t \mid \mathbf{y}_{1:t})$, $\mathbf{X}_t = \mathbf{x}_t^1,...,\mathbf{x}_t^N$. To compute the above distribution function, some remarks should be first noted as

1. That how to assign each target state to a measurement is crucial. Indeed, at each time step the sensor provides a set of measurements. The source of these measurements can be the targets or the disturbances also known as clutters. Therefore, a special procedure is needed to assign each target to its associated measurement. This procedure is designated as Data Association considered as a key stage of the JPDAF algorithm which is described in next sections.
2. Because the JPDAF algorithm updates the estimated states sequentially, a recursive solution should be applied to update the states at each sample time. Traditionally, Kalman filtering has been a strong tool for recursively estimating the states of the targets in the multi-target tracking scenario. Recently, particle filters joint with the data association strategy have provided better estimations, specially, when the sensor model is nonlinear.

With regard to the above points, the following sections describe how particle filters paralleled with the data association concept can deal with the multi-target tracking problem.

2.1 The particle filter for online state estimation

Consider the problem of online state estimation as computing the posterior probability density function $p(\mathbf{x}_t^k \mid \mathbf{y}_{1:t})$. To provide a recursive formulation for computing the above density function, the following stages are presented:

1. *Prediction stage*: the prediction step is proceeded independently for each target as

$$p(\mathbf{x}_t^k \mid \mathbf{y}_{1:t-1}) = \int_{\mathbf{x}_{t-1}^k} p(\mathbf{x}_t^k \mid \mathbf{x}_{t-1}^k) p(\mathbf{x}_{t-1}^k \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}^k \tag{2}$$

2. *Update stage*: this step can be also described as follows:

$$p(\mathrm{x}_t^k \mid \mathrm{y}_{1:t}) \propto p(\mathrm{y}_t \mid \mathrm{x}_t^k) p(\mathrm{x}_t^k \mid \mathrm{y}_{1:t-1}) \tag{3}$$

The particle filter algorithm estimates the probability distribution density function $p(\mathbf{x}_t^k \mid \mathbf{y}_{1:t-1})$ by sampling from a specific distribution function as follows:

$$p(\mathbf{x}_t^k \mid \mathbf{y}_{1:t-1}) = \sum_{i=1}^{N} \tilde{w}_t^i \delta(\mathbf{x}_t - \mathbf{x}_t^i) \tag{4}$$

Here i=1,2,...,N is the sample number, $\tilde{\mathbf{w}}_t$ is the normalized importance weight and $\delta(.)$ is the delta dirac function. In the above equation, the state $\mathbf{x}_t^i$ is sampled from the proposal density function $q(\mathbf{x}_t^k \mid \mathbf{x}_{t-1}^k, \mathbf{y}_{1:t})$. By substituting the above equation in (2) and the fact that states are drawn from the proposal function q, the recursive equation for the prediction step can be written as follows:

$$\alpha_t^i = \alpha_{t-1}^i \frac{p(\mathbf{x}_t^k \mid \mathbf{x}_{t-1}^{k,i})}{q(\mathbf{x}_t^k \mid \mathbf{x}_{t-1}^{k,i}, \mathbf{y}_{1:t})} \tag{5}$$

where $\mathbf{x}_{t-1}^{k,i}$ is the i^{th} sample of $\mathbf{x}_{t-1}^k$. Now, by using (3) the update stage can be expressed as a recursive adjustment of importance weights as follows:

$$\mathbf{w}_t^i = \alpha_t^i p(\mathbf{y}_t \mid \mathbf{x}_t^k) \tag{6}$$

By repeating the above procedure at each time step, the sequential importance sampling (SIS) algorithm for online state estimation is presented as below:

```
1. For i=1: N initialize the states $\mathbf{x}_0^i$, prediction weights $\alpha_0^i$ and importance weights $w_0^i$.
2. At each time step t proceed the following stages:
    a. Sample states from the proposal density function as follows:
           $\mathbf{x}_t^i \sim q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^i, \mathbf{y}_{1:t})$        (7)
    b. Update prediction weights by (5).
    c. Update importance weights by (6).
    d. Normalize importance weights as follows:
           $\tilde{w}_t^i = \frac{w_t^i}{\sum_{i=1}^{N} w_t^i}$        (8)
3. Set t=t+1 and go to 2.
```

Table. 1. The SIS Algorithm

For the sake of simplicity, the index k has been eliminated in Table 1. The main failure of the SIS algorithm is the degeneracy problem. That is, after a few iterations one of the normalized importance ratios tends to 1, while the remaining ratios tend to zero. This problem causes the variance of the importance weights to increase stochastically over time (Del Moral et al., 2006). To avoid the degeneracy of the SIS algorithm, a selection (resampling) stage may be used to eliminate samples with low importance weights and multiply samples with high importance weights. There are many approaches to implement the resampling stage (Del Moral et al., 2006). Among them, the residual resampling provides a straightforward strategy to solve the degeneracy problem in the SIS algorithm. By combining the concept of residual resampling with the SIS algorithm presented before, the SIR algorithm is described in Table 2.

```
1. For i=1: N initialize the states $\mathbf{x}_0^i$, prediction weights $\alpha_0^i$ and importance weights $w_0^i$.
2. At each time step t do the following stages:
   a. Do the SIS algorithm to sample states $\mathbf{x}_t^i$ and compute normalized importance weights $\tilde{w}_t^i$.
   b. Check the resampling criterion:
      i. If $N_{eff} > thresh$, follow the SIS algorithm Else:
      ii. Implement the residual resampling stage to multiply/suppress $\mathbf{x}_t^i$ with high/low importance weights.
      iii. Set the new normalized importance weights as $\tilde{w}_t^i = \frac{1}{N}$.
3. Set t=t+1 and go to 2.
```

Table. 2. The SIR Algorithm

In the above algorithm, N_{eff} is a criterion checking the degeneracy problem which can be written as:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (\tilde{w}_t^i)^2} \tag{9}$$

In (Freitas 1999), a comprehensive discussion has been made on how one can implement the residual resampling stage.

Besides the SIR algorithm, some other approaches have been proposed in the literature to enhance the quality of the SIR algorithm such as Markov Chain Monte Carlo particle filters, Hybrid SIR and auxiliary particle filters (Freitas 1999). Although these methods are more accurate than the common SIR algorithm, some other problems such as the computational cost are the most significant reasons that, in many real time applications such as online tracking, the traditional SIR algorithm is applied to recursive state estimation.

2.2 Data association

In the last section, the SIR algorithm was presented for online state estimation. However, the major problem of the proposed algorithm is how to compute the likelihood function $p(\mathbf{y}_t \mid \mathbf{x}_t^k)$. To do so, an association should be made between measurements and targets. Generally, two types of association may be defined as follows:

Definition 1: we will denote a target to measurement association ($T \to M$) by $\tilde{\lambda} = \{\tilde{r}, m_c, m_T\}$ where $\tilde{r} = \{\tilde{r}_1, ..., \tilde{r}_K\}$ and $\tilde{r}_k$ is defined as:

$$\tilde{r}_k = \begin{cases} 0 & \textit{If the } k^{th} \textit{ target is not detected} \\ j & \textit{f the } k^{th} \textit{ target has generated the } j^{th} \textit{ measurement} \end{cases} \tag{10}$$

where j=1,2,...,m and m is the number of measurements at each time step and k=1,2,..,K which K is the number of targets.

Definition 2: in a similar fashion, the measurement to target association ($M \to T$) is defined as $\lambda = \{r, m_c, m_T\}$ where $r = \{r_1, ..., r_m\}$ and r_j is defined as follows:

$$\tilde{r}_j = \begin{cases} 0 & \textit{If the } j^{th} \textit{ measurement is not associated} \\ k & \textit{if the } j^{th} \textit{ measurement is associated to the } k^{th} \textit{ target} \end{cases} \tag{11}$$

In both above equations, m_T is the number of measurements due to the targets and m_c is the number of clutters. It is very easy to show that both definitions are equivalent but the dimension of the target to measurement association is less than the measurement to target association dimension. Therefore, in this paper, the target to measurement association is used. Now, the likelihood function for each target can be written as

$$p(\mathbf{y}_t \mid \mathbf{x}_t^k) = \beta^0 + \sum_{i=1}^{m} \beta^i p(\mathbf{y}_t^i \mid \mathbf{x}_t^k) \tag{12}$$

In the above equation, β^i is defined as the probability that the i^{th} measurement is assigned to the k^{th} target. Therefore, β^i can be written as the following equation:

$$\beta^i = p(\tilde{r}_k = i \mid \mathbf{y}_{1:t}) \tag{13}$$

Before describing how to compute the above equation, the following definition is represented:

Definition 3: we define the set $\tilde{\lambda}$ as all possible assignments which can be made between measurements and targets. For example, consider a 3-target tracking problem. Assume that the following associations are recognized between targets and measurements:
$r_1 = \{1,2\}, r_2 = \{3\}, r_3 = \{0\}$. Now, the set $\tilde{\lambda}$ can be shown in Table 3.

Target 1	Target 2	Target 3
0	0	0
0	3	0
1	0	0
1	3	0
2	0	0
2	3	0

Table. 3. All possible associations between the targets and measurements for example 1

In Table 3, 0 means that the target has not been detected. By using the above definition, (11) can be rewritten as

$$p(\tilde{r}_k = i \mid \mathbf{y}_{1:t}) = \sum_{\tilde{\lambda}_t \in \tilde{\Lambda}_t, \tilde{r}_k = j} p(\tilde{\lambda}_t \mid \mathbf{y}_{1:t}) \tag{14}$$

The right side of the above equation is written as follows:

$$p(\tilde{\lambda}_t \mid \mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \tilde{\lambda}_t) p(\mathbf{y}_{1:t-1}, \tilde{\lambda}_t)}{p(\mathbf{y}_{1:t})} = p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \tilde{\lambda}_t) p(\tilde{\lambda}_t) \tag{15}$$

In the above equation, we have used the fact that the association vector $\tilde{\lambda}_t$ is not dependent on the history of measurements. Each of density functions in the right hand side of the above equation can be computed as follows:

1. $p(\tilde{\lambda}_t)$

The above density function can be written as

$$p(\tilde{\lambda}_t) = p(\tilde{r}_t, m_c, m_T) = p(\tilde{r}_t \mid m_c, m_T) p(m_c) p(m_T) \tag{16}$$

The computation of each density function in the right hand side of the above equation is straightforward and can be found in (Freitas 1999).

2. $p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \tilde{\lambda}_t)$

Because targets are assumed to be independent, the above density function can be written as follows:

$$p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \tilde{\lambda}_t) = (V_{\max})^{-m_c} \Pi_{j \in \Gamma} p^{r_t^j} (\mathbf{y}_t^j \mid \mathbf{y}_{1:t-1}) \tag{16}$$

where $V_{\max}$ is the maximum volume in the line of sight of sensors, Γ is the set of all valid measurement to target associations and $p^{r_t^j} (\mathbf{y}_t^j \mid \mathbf{y}_{1:t-1})$ is the predictive likelihood for the $(r_t^j)^{th}$ target. Now, consider $r_t^j = k$. The predictive likelihood function for the k^{th} target can be written as follows:

$$p^k (\mathbf{y}_t^j \mid \mathbf{y}_{1:t-1}) = \int p(\mathbf{y}_t^j \mid \mathbf{x}_t^k) p(\mathbf{x}_t^k \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_t^k \tag{17}$$

Both density functions in the right hand side of the above equation are estimated using the samples drawn from the proposal density function. However, the main problem is how to determine the association between measurements $\mathbf{y}_t$ and targets. To do so, the soft gating method is proposed in Table 4.

```
1. Consider $\mathbf{x}_t^{i,k}$, i=1,2,..,N, as the samples drawn from the proposal density function.
2. For j=1:m do the following steps for the $j^{th}$ measurement:
   a. For k=1:K do the following steps:
      i. Compute $\mu_j^k$ as follows:
            $$\mu_j^k = \sum_{i=1}^{N} \tilde{\alpha}_t^{i,k} g(\mathbf{x}_t^{i,k}) \quad (18)$$
         Where g is the sensor model and $\tilde{\alpha}_t^k$ is the normalized weight as presented
         in the SIR algorithm.
      ii. Compute $\sigma_j^k$ by the following equation:
            $$\sigma_j^k = R + \sum_{i=1}^{N} \tilde{\alpha}_t^{i,k} \left(g(\mathbf{x}_t^{i,k}) - \mu_j^k\right)\left(g(\mathbf{x}_t^{i,k}) - \mu_j^k\right)^T \quad (19)$$
      iii. Compute the distance to the $j^{th}$ target as follows:
            $$d_j^2 = \frac{1}{2}(\mathbf{y}_t^j - \mu_j^k)^T (\sigma_j^k)^{-1} (\mathbf{y}_t^j - \mu_j^k) \quad (20)$$
      iv. If $d_j^2 < \varepsilon$, assign the $j^{th}$ measurement to the $k^{th}$ target.
   b. End of the loop for k.
3. End of the loop for j.
```

Table. 4. Soft gating for data association

It is easy to show that the predictive likelihood function presented in (16) can be approximated as follows:

$$p^k(\mathbf{y}_t^j \mid \mathbf{y}_{1:t-1}) \approx N(\mu^k, \sigma^k) \qquad (21)$$

where $N(\mu^k, \sigma^k)$ is a normal distribution with mean μ^k and the covariance matrix σ^k. By computing the predictive likelihood function, the likelihood density function can be easily estimated. In the next subsection, the JPDAF algorithm is presented for multi-target tracking.

2.3 The JPDAF algorithm

The mathematical foundations of the JPDAF algorithm were discussed in the last sections. Now, we are ready to propose the full JPDAF algorithm for the problem of multiple target tracking. To do so, each target is characterized by a dynamical model introduced by (1). The JPDAF algorithm is, therefore, presented in Table 5.

1. **Initialization:** initialize states for each target as $\mathbf{x}_0^{i,k}$ where i=1,2,..,N and k=1,2,...,K, the predictive importance weights $\alpha_0^{i,k}$ importance weights $w_0^{i,k}$.
2. At each time step t proceed through the following stages:
   - a. For k=1:K conduct the following procedures for each target:
     - i. For i=1:N do the following steps:
     - ii. Sample the new states from the proposal density function as follows:
       $$\mathbf{x}_t^{i,k} \sim q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{i,k}, \mathbf{y}_{1:t}) \tag{22}$$
     - iii. Update the predictive importance weights as
       $$\alpha_t^{i,k} = \alpha_{t-1}^{i,k} \frac{p(\mathbf{x}_t^{i,k} \mid \mathbf{x}_{t-1}^{i,k})}{q(\mathbf{x}_t^{i,k} \mid \mathbf{x}_{t-1}^{i,k}, \mathbf{y}_{1:t})} \tag{23}$$
       Then, normalize the predictive importance weights.
     - iv. Use the sampled states and new observations $\mathbf{y}_t$ to constitute the association vector for each target as $R^k = \{j \mid 0 \le j \le m, \mathbf{y}_j \to k\}$ where ($\to k$) refers to the association between the k^{th} target and the j^{th} measurement. Use the soft gating procedure described in the last subsection to compute each association.
     - v. Constitute all possible associations for the targets and make the set $\tilde{\Gamma}$ as described in Definition 3.
     - vi. Use (13) and compute β^l for each measurement where l=1,2,..,m and m is the number of measurements.
     - vii. By using (10) compute the likelihood ratio for each target as $p(\mathbf{y}_t \mid \mathbf{x}_t^{i,k})$.
     - viii. Compute importance weights and normalize them as follows:
       $$w_t^{i,k} = \alpha_t^{i,k} p(\mathbf{y}_t \mid \mathbf{x}_t^{i,k}),\ \tilde{w}_t^{i,k} = \frac{w_t^{i,k}}{\sum_{i=1}^{N} w_t^{i,k}} \tag{24}$$
     - ix. Implement the resampling stage. To do so, do the similar procedure described in the SIR algorithm. Afterwards, for each target the resampled states can be presented as follows:
       $$\{\tilde{w}_t^{i,k}, \mathbf{x}_t^{i,k}\} \to \{\frac{1}{N}, \mathbf{x}_t^{m(i),k}\} \tag{25}$$
   - b. End of the loop for k.
3. Set t=t+1 and go to step 2.

Table. 5. The JPDAF algorithm for multiple target tracking

The above algorithm can be used to deal with the multi-target tracking scenario. In the next section, we show how the JPDAF algorithm can be used for multiple robot tracking. In addition, some well-known motion models are presented to track the motion of a mobile robot.

3. The JPDAF algorithm for multiple robot tracking

In the last section, the JPDAF algorithm was completely discussed. Now, we want to use the JPDAF algorithm for the problem of multiple robot tracking. To do so, consider a simple 2-wheel differential mobile robot, Fig. 1, whose dynamical model is represented as follows:

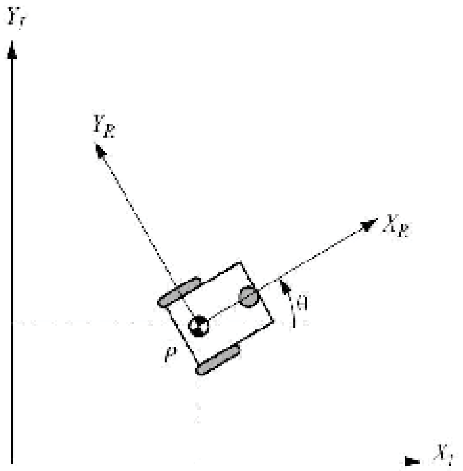


Fig. 1. A 2-wheel differential mobile robot

$$x_{t+1} = x_t + \frac{\Delta s_r + \Delta s_l}{2}\cos(\theta_t + \frac{\Delta s_r - \Delta s_l}{2b})$$

$$y_{t+1} = y_t + \frac{\Delta s_r + \Delta s_l}{2}\sin(\theta_t + \frac{\Delta s_r - \Delta s_l}{2b})$$

$$\theta_{t+1} = \theta_t + \frac{\Delta s_r - \Delta s_l}{b} \quad (26)$$

where $[x_t, y_t]$ is the position of the robot, θ_t is the angle of the robot's head, Δs_r and Δs_l are the distances travelled by each wheel, and b refers to the distance between two wheels of the robot. The above equation describes a simple model presenting the motion of a differential mobile robot. For a single mobile robot localization problem, the most straightforward way is to use this model and the data collected from the sensors set on the left and right wheels measuring Δs_r and Δs_l at each time step. But, the above method does not lead to the suitable results because the data gathered from the sensors are dotted with the additive noise and, therefore, the estimated trajectory does not match the actual trajectory. To remedy this problem, measurements obtained from the sensors are used to modify the estimated states. Therefore, Kalman and particle filters have been greatly applied to the problem of mobile robot localization (Siegwart & Nourbakhsh, 2004). Now, consider the case in which the position of other robots should be identified by a reference robot. In this situation, the dynamical model discussed previously is not applicable because the reference robot does not have any access to the internal sensors of other robots such as the sensors measuring the movement of each wheel. Therefore, a motion model should be first defined for the movement of each mobile robot. The simplest model is a near constant velocity model presented as follows:

$$\begin{aligned} \mathbf{X}_{t+1} &= A\mathbf{X}_t + B\mathbf{u}_t \\ \mathbf{X}_t &= [x_t, \dot{x}_t, y_t, \dot{y}_t] \end{aligned} \tag{27}$$

where the system matrixes are defined as

$$A = \begin{pmatrix} 1 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 \end{pmatrix}, B = \begin{pmatrix} \frac{T_s^2}{2} & 0 \\ T_s & 0 \\ 0 & \frac{T_s^2}{2} \\ 0 & T_s \end{pmatrix} \tag{28}$$

where T_s refers to the sample time. In the above equations, $\mathbf{u}_t$ is a white noise with zero mean and an arbitrary covariance matrix. Because the model is supposed to be a near constant velocity model, the covariance of the additive noise should not be so large. Indeed, this model is suitable for the movement of targets with a constant velocity which is common in many applications such as aircraft path tracking and people tracking.

The movement of a mobile robot can be described by the above model in many conditions. However, in some special cases the robots' movement can not be characterized by a simple near constant velocity model. For example, in a robot soccer problem, the robots may conduct a manoeuvring movement to reach a special target such as a ball. In these cases, the robot changes its orientation by varying input forces imposed to the right and left wheels. Therefore, the motion trajectory of the robot is so complex that an elementary constant velocity model may not result in satisfactory responses. To overcome the mentioned problem, the variable velocity model is proposed. The key idea behind this model is using the robot's acceleration as another state variable which should be estimated as well as the velocity and position of the robot. Therefore, the new state vector is defined as $\mathbf{X}_t = [x_t, \dot{x}_t, a_t^x, y_t, \dot{y}_t, a_t^y]$ where a_t^x, a_t^y are the robot's acceleration along with the x and y axis, respectively. Now, the near constant acceleration model can be written similar to what mentioned in (27), except for the system matrixes which are defined as follows (Ikoma et all., 2001):

$$A = \begin{pmatrix} 1 & 0 & T_s & 0 & a_1 & 0 \\ 0 & 1 & 0 & T_s & 0 & a_1 \\ 0 & 0 & 1 & 0 & a_2 & 0 \\ 0 & 0 & 0 & 1 & 0 & a_2 \\ 0 & 0 & 0 & 0 & \exp(-T_s) & 0 \\ 0 & 0 & 0 & 0 & 0 & \exp(-T_s) \end{pmatrix}, B = \begin{pmatrix} b_1 & 0 \\ 0 & b_1 \\ b_2 & 0 \\ 0 & b_2 \\ b_3 & 0 \\ 0 & b_3 \end{pmatrix} \tag{30}$$

where the constants of the above equation are defined as

$$
\begin{aligned}
&b_3 = \frac{1}{c}\left(1-\exp(-cT_s)\right), a_2 = b_3, b_2 = \frac{1}{c}\left(T_s - a_2\right) \\
&a_1 = b_2, b_1 = \frac{1}{c}\left(\frac{T_s^2}{2} - a_1\right)
\end{aligned}
\tag{31}
$$

In the above equation, c is a constant value. The above model can be used to track the motion trajectory of a manoeuvring object, such as the movement of a mobile robot. Moreover, using the results of the proposed models can enhance the performance of tracking. This idea can be presented as follows:

$$
\tilde{\mathbf{X}}_t = \alpha\tilde{\mathbf{X}}_t^a + (1-\alpha)\tilde{\mathbf{X}}_t^v \tag{32}
$$

where $\tilde{\mathbf{X}}_t^a$ and $\tilde{\mathbf{X}}_t^v$ are the estimation of the near constant acceleration and near constant velocity model, respectively, and α is a number between 0 and 1. To adjust α, an adaptive method is used considering the number of measurements assigned to targets when each model is used separately. That is, more the number of measurements is assigned to targets by each model, the larger value is chosen for α.

Besides the above approaches, some other methods have been proposed to deal with tracking of manoeuvring objects such as Interactive Multiple Mode (IMM) filters (Pitre et al., 2005). However, these methods are applied to track the movement of objects with sudden manoeuvring movement which is common in aircrafts. In the mobile robots scenario, the robot's motion is not so erratic that the above method is necessary. Therefore, the near constant velocity model, near constant acceleration model or a combination of the proposed models can be considered as the most suitable structures which can be used to represent the dynamical model of a mobile robot. Afterwards, the JPDAF algorithm can be easily applied to the multi-robot tracking problem.

4. A fuzzy controller for optimal observer trajectory planning

In this section, a novel fuzzy logic controller (FLC) is proposed to maintain a better tracking quality for the multi-robot tracking problem. Indeed, the major motivation of using a moving platform for the reference robot is some weaknesses in the traditional multiple target tracking scenarios in which sensors were assumed to be fixed or conduct an independent movement from mobile targets. The most important weaknesses found in recent approaches are:

- Generally, the variance of the additive noise of sensors increases when the distance between each sensor and mobile targets increases. Consequently, the quality of tracking will decrease if the position of the sensor or reference robot is fixed.
- If the position of the reference robot is assumed to be fixed, targets may exit the reference robot's field of view and, therefore, the reference robot will lose other robots' position.
- In some applications, the reference robot may be required to track a special target. This problem is common in some topics such as the robot rescue or security planning, when the reference robot should follow the movement of an individual. The traditional approaches are not flexible enough to be applied for the path/trajectory following problem.

- Switching between trajectory tracking and path following is either impossible or very hard and time consuming when traditional approaches are used, when the reference robot is fixed or its movement is independent from other robots' movement.

To remedy aforementioned flaws, a novel strategy is proposed to provide a trajectory for the reference robot dependent on other targets' movement. To obtain an optimal solution, the following cost function is defined for the trajectory planning problem:

$$R = \sqrt{\frac{\sum_{i=1}^{n} (r_i)^2}{n}} \qquad \Phi = \sqrt{\frac{\sum_{i=1}^{n} (\varphi_i)^2}{n}} \tag{33}$$

where r_i is the distance between the i^{th} target and reference robot and φ_i is the angle between the i^{th} target and reference robot in the local coordination of the reference robot (Fig. 2).

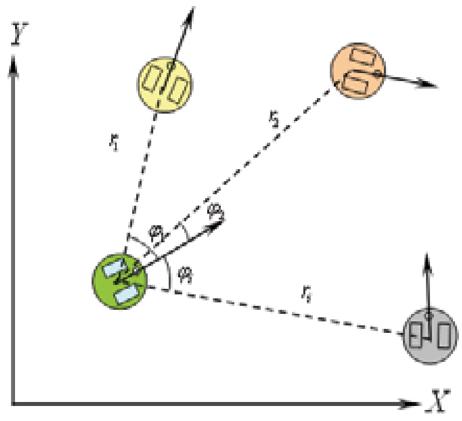


Fig. 2. Position of targets in the reference robot's coordination

The major reason for defining the above cost function is that, by this way, the reference robot maintains its minimal distance to all targets and, thus, the performance of the target tracking is improved. In other words, when the reference robot is placed in a position whose sum of distances to other robots is minimal, the effect of the additive noise dependent on the distance between targets and the reference robot is also minimized. Therefore, the trajectory planning problem is defined as a method causing the reference robot to move along a suitable path minimizing the pre-mentioned cost function.

In order to execute the aforementioned trajectory planning problem, a robust fuzzy controller is used. Fig. 3 shows the block diagram of the proposed fuzzy controller. In design of fuzzy logic controllers, we use the Mamdani type of the fuzzy control containing fuzzification and defuzzification stages and, also, a rule base. The task of the fuzzy controller is to have the reference robot follow the above-discussed optimal trajectory smoothly and, of course, precisely. In this paper, we use an FLC based on kinematical model of the robot.

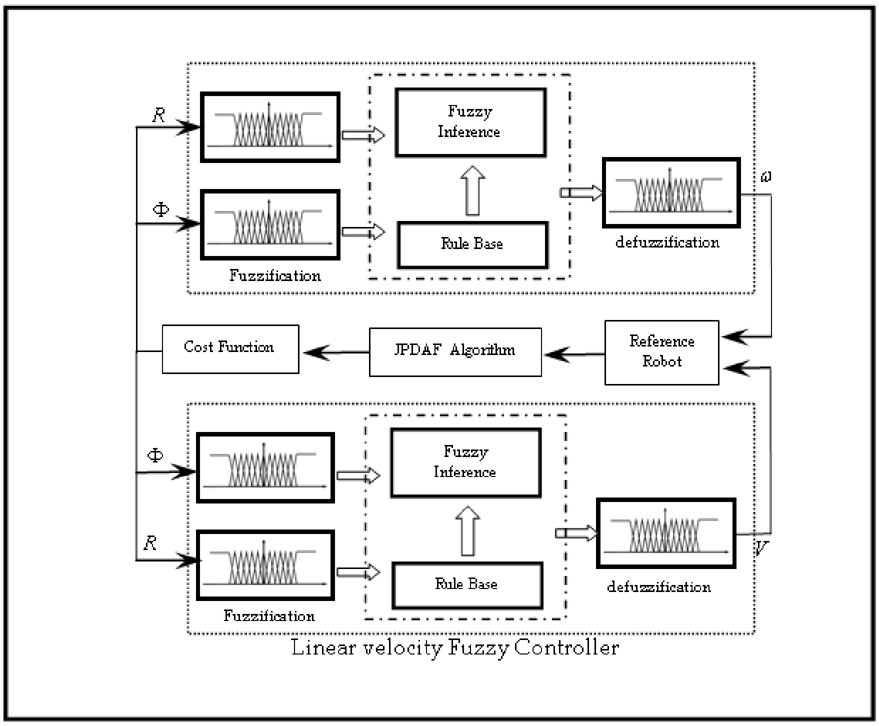


Fig. 3. A block diagram of represented fuzzy controller

After testing various number of membership functions for input variables R, Φ, the best fuzzy system for the angular velocity control is designed with *seven* triangular membership functions. Although there is no restriction on the general form of membership functions, we choose the piecewise linear description (Fig. 4 & Fig. 5).

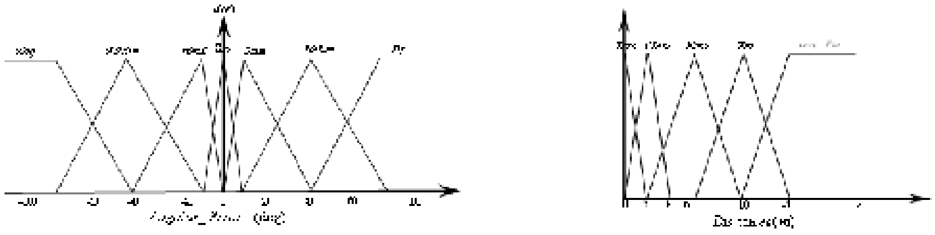


Fig. 4. Membership functions of input fuzzy sets of fuzzy controllers

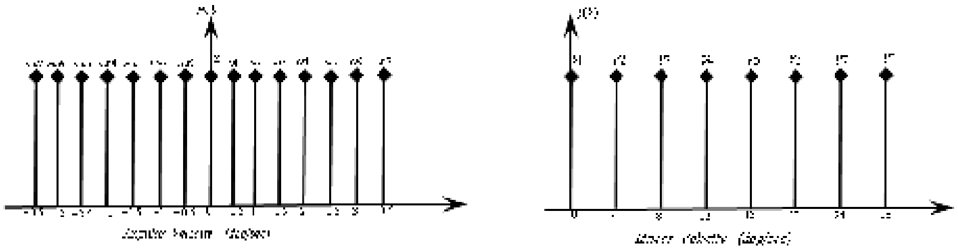


Fig. 5. Membership functions of output fuzzy sets of fuzzy controllers

The second step in designing an FLC is the fuzzy inference mechanism. The knowledge base of the angular and linear velocity of fuzzy controllers consists of the rules described in Table 6 & Table 7.

Big	**Medium**	**Small**	**Zero**	**nSmall**	**nMedium**	**nBig**	R \ Φ
Ω7	ω6	Ω5	Z	nω5	nω6	nω7	**Zero**
Ω6	ω5	Ω4	Z	nω4	nω5	nω6	**Close**
Ω5	ω4	Ω3	Z	nω3	nω4	nω5	**Near**
Ω4	ω3	Ω2	Z	nω2	nω3	nω4	**Far**
Ω3	ω2	Ω1	Z	nω1	nω2	nω3	**very_Far**

Table. 6. The fuzzy rule base for the angular velocity fuzzy controller

Big	**Medium**	**Small**	**Zero**	**nSmall**	**nMedium**	**nBig**	R \ Φ
V1	V2	V3	V4	V3	V2	V1	**Zero**
V2	V3	V4	V5	V4	V3	V2	**Close**
V3	V4	V5	V6	V5	V4	V3	**Near**
V4	V5	V6	V7	V6	V5	V4	**Far**
V5	V6	V7	V8	V7	V6	V5	**very_Far**

Table. 7. The fuzzy rule base for the linear velocity fuzzy controller

In this application, an algebraic product is used for all of the *t-norm* operators, max is used for all of the *s-norm*, as well as individual-rule based inference with union combination and mamdani's product implication . Product inference engine is defined as

$$\mu_{B'}(y) = \max_{l=1}^{M}\left[\sup_{x\in U}\left(\mu_{A'}(x)\prod_{i=1}^{n}\mu_{A^l_i}(x_i)\mu_{B^l}(y)\right)\right] \tag{34}$$

There are many alternatives to perform the defuzzification stage. The strategy adopted here is the Center Average defuzzification method. This method is simple and very quick and can be implemented by (Wang 1997)

$$y^* = \frac{\sum_{l=1}^{M}\bar{y}^l w_l}{\sum_{l=1}^{M} w_l} \tag{35}$$

where $\bar{y}^l$ is the center of th l^{th} fuzzy set and w_l is its heigh.

The above procedures provide a strong tool for designing a suitable controller for the reference robot leading to a better tracking performance. The next section shows how this approach enhances the accuarcy of tracking.

5. Simulation results

To evaluate the efficiency of the JPDAF algorithm, a 3- robot tracking scenario is designed. Fig. 6 shows the initial position of the reference and target robots. To prepare the scenario and implement the simulations, parameters are first determined for the mobile robots' structure and simulation environment by Table 8. Now, the following experiments are conducted to evaluate the performance of the JPDAF algorithm in various situations.

Parameters	Description
v_l	The robot's left wheel speed
v_r	The robot's right wheel speed
b	The distance between the robot's wheels
n_s	Number of sensors placed on each robot
$R_{\max}$	Maximum range of simulation environment
Q_s	The covariance matrix of the sensors' noise
t_s	Sample time for simulation
$t_{\max}$	Maximum running time for the simulation

Table. 8. Simulation parameters

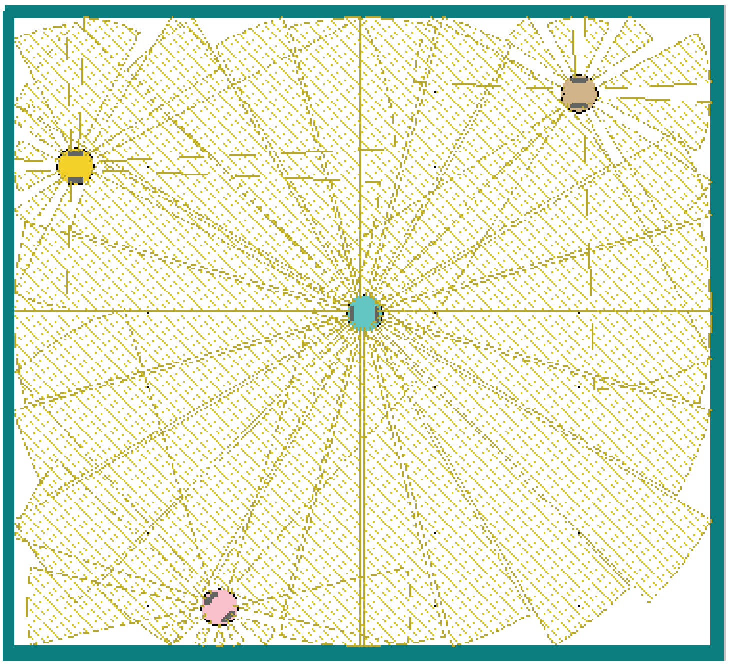


Fig. 6. The generated trajectory for mobile robots in a simulated environment

5.1 Local tracking for manoeuvring movement and the fixed reference robot

To design a suitable scenario, the speed of each robot's wheel is determined by Table 3. The initial position of the reference robot is set in [0, 0, 0]. Moreover, other robots' position is assumed to be [2, 2, π], [1, 3, $\frac{\pi}{3}$], [1, 9, $\frac{\pi}{4}$], respectively. To run the simulation, sample time t_s and maximum running time are also set in 1s and 200s, respectively. To consider the uncertainty in measurements provided by sensors, a Gaussian noise with zero mean and the covariance matrix $\begin{pmatrix} .2 & 0 \\ 0 & 5\times 10^{-4} \end{pmatrix}$ is added to measurements obtained by sensors. After simulating the above-mentioned 3-robot scenario with aforementioned parameters, the generated trajectories can be observed in Fig. 7.

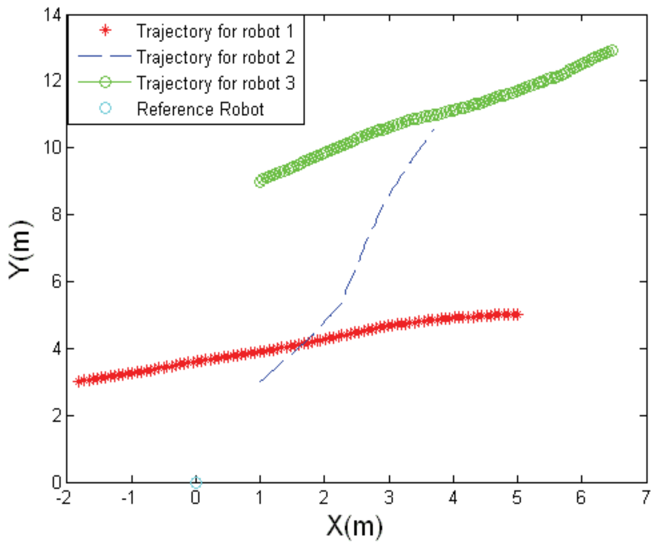
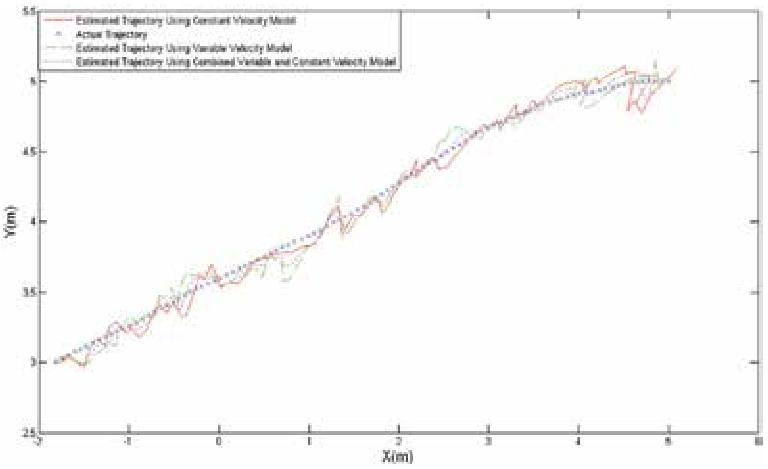


Fig. 7. Generated trajectories for multi-robot tracking scenario

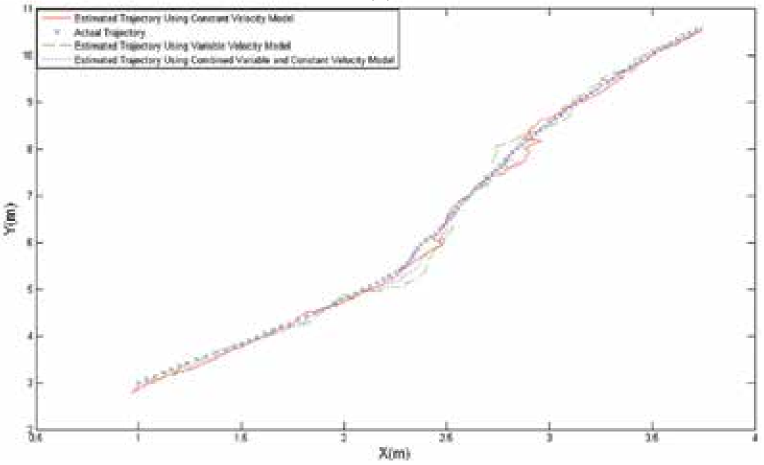
Now, we are ready to implement the JPDAF algorithm discussed before. First, a JPDAF algorithm with 500 particles is used to track the each target's movement. To compare the efficiency of each motion model described in section 3, simulations are done for different motion models. Fig. 8 shows the tracking results for various models and targets. To provide a better view to the accuracy of each approach, Table 9 and Table 10 present the tracking error for each model where the following criterion is used to compute the error:

$$e_z = \frac{\sum_{t=1}^{t_{\max}} (z_t - \tilde{z}_t)^2}{t_{\max}}, z_t = \{x_t, y_t\} \tag{35}$$

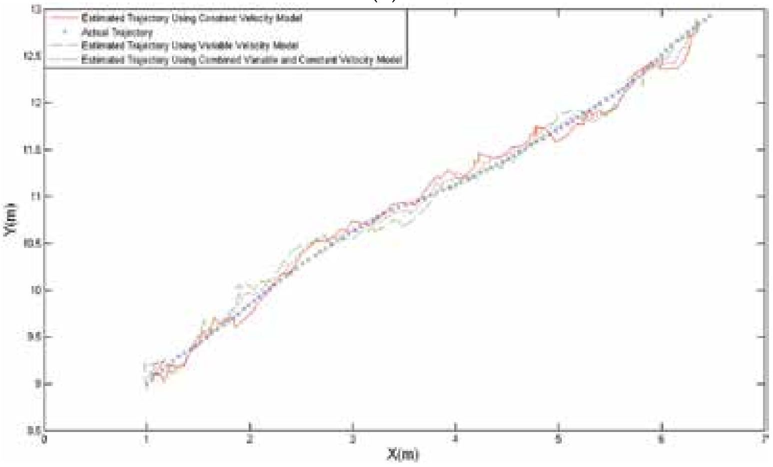
From Tables, it is obvious that the combined model has resulted in a better performance than other motion models. Indeed, near constant velocity and acceleration models do not provide similar results during the simulation and, therefore, the combined model mixing results obtained from each model has led to much better performance.



(a)



(b)



(c)

Fig. 8. Tracking results using the JPDAF algorithm with 500 particles for all robots

Robot Number	Const. Velocity	Const. Acceleration	Combined Model
1	0.002329	0.006355	0.002549
2	0.003494	0.003602	0.001691
3	0.005642	0.005585	0.002716

Table. 9. Tracking errors of estimating x_t for various motion models

Robot Number	Const. Velocity	Const. Acceleration	Combined Model
1	0.005834	0.006066	0.003755
2	0.016383	0.009528	0.007546
3	0.011708	0.012067	0.006446

Table. 10. Tracking errors of estimating y_t for various motion models

5.2 Local tracking using a mobile reference robot

Now, we apply the control strategy presented in section 4 for finding an optimal trajectory for the reference robot. To implement the simulation, the reference robot is placed at [0, 0]. A fuzzy controller with two outputs is designed to find the linear and angular velocity of the reference robot. Simulations are conducted with parameters similar to ones defined in the last section. Moreover, to consider the effect of the distance between sensors and targets, the covariance of the additive noise is varied by changing the distance. Fig. 9 explains how we have modelled the uncertainty in measurements received by sensors.

After running the simulation for 400s, the fuzzy controller finds a trajectory for the mobile robot. Fig. 10 shows the obtained trajectory after applying the JPDAF algorithm with 500 particles for finding the position of each mobile target.

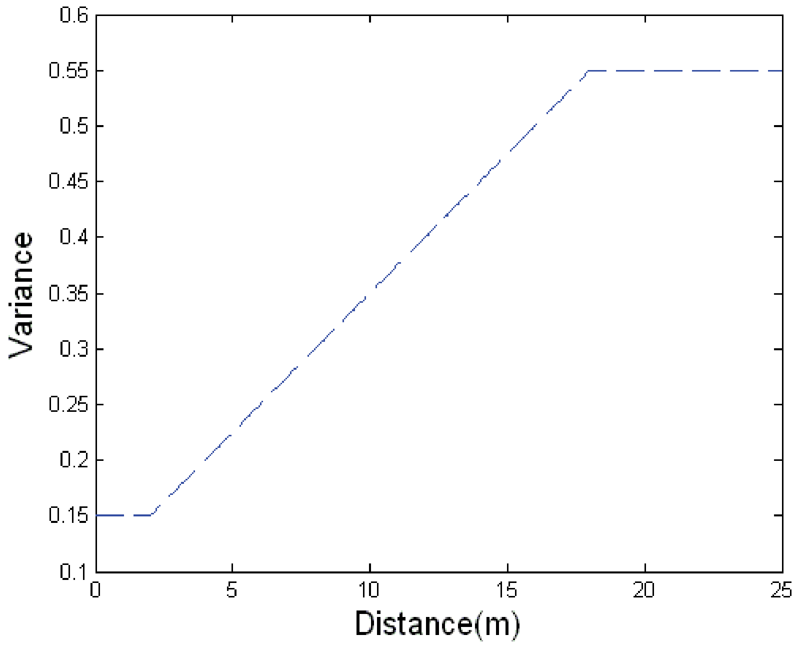


Fig. 9. A model used to describe the behaviour of additive noises by changing the distance

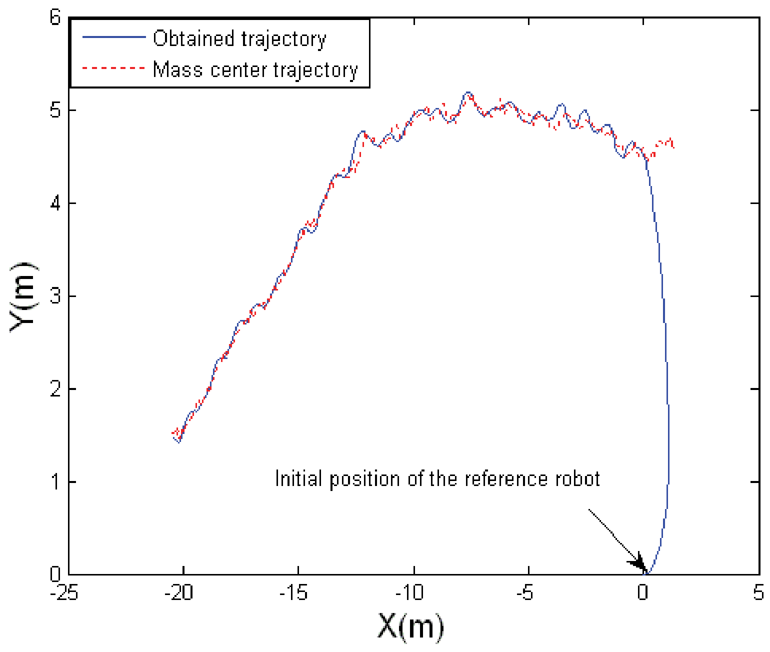
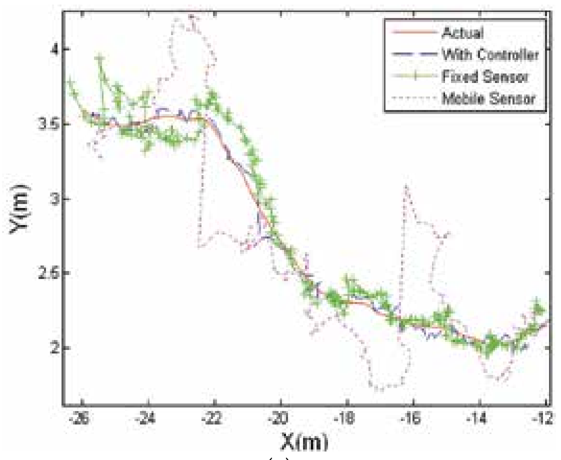
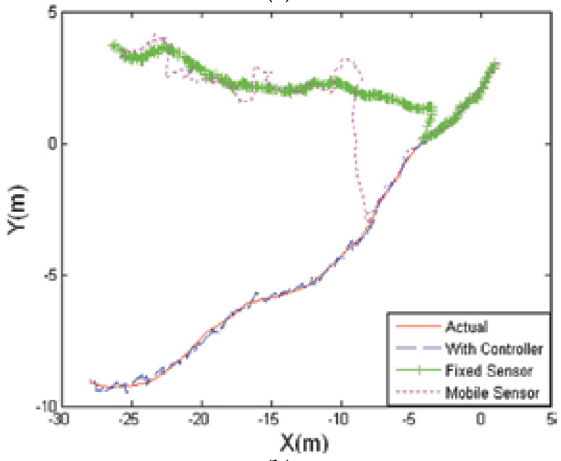


Fig. 10. The obtained trajectory for the reference robot using the fuzzy controller

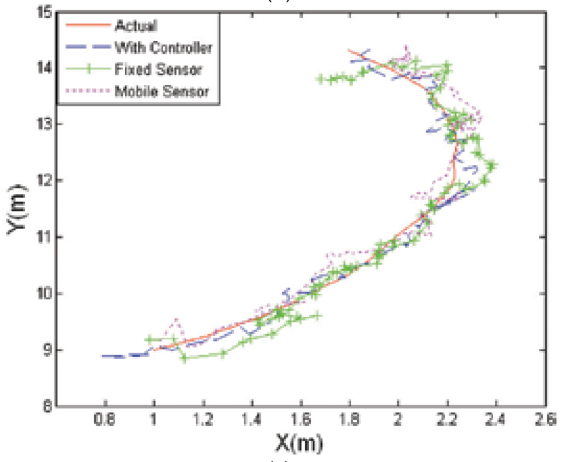
The above figure justifies this fact that the reference robot has been directed to the geometrical placement of the targets' mass centre. Additionally, the simulation is run for situations in which the reference robot is fixed and, also, mobile with an arbitrary trajectory, without applying the control scheme. Fig. 11 shows tracking results using each of the above-mentioned strategies. Also, tables 11 & 12 present the estimated tracking error for all cases. Simulation results show the better tracking performance after applying the control strategy. In other words, because the controller directs the reference robot to a path in which the sum of the distance to other robots is minimal, the tracking algorithm is able to find other robots' position much better than the case in which the reference robot is fixed. In addition, when the reference robot is moving without any special plan, or, at least, its movement is completely independent from other robots' movement, the tracking performance appears much worse. The above-mentioned evidence shows that the effect of the additive noise's variance is so much that the reference robot has completely lost the trajectory of the target shown in Fig. 11 (c). In this case, using the control strategy has caused the reference robot to be placed in a position equally far from other robots and, therefore, the effect of the additive noise weakens. The most important advantage of our proposed approach compared with other methods suggested in the literature for observer trajectory planning (Singh et al., 2007) is that the fuzzy controller can be used in an online mode while recent approaches are more applicable in offline themes. The aforementioned benefit causes our approach to be easily applied to many real topics such as robot rescue, simultaneous localization and mapping (SLAM), etc.



(a)



(b)



(c)

Fig. 11. Tracking results for various situations of the reference robot (fixed, mobile with and without a controller)

Robot Number	Fixed Ref. Robot	Mobile Ref. Robot	Ref. Robot With Controller
1	0.024857	0.077568	0.005146
2	0.883803	0.821963	0.005236
3	0.009525	0.025623	0.009882

Table. 11. Tracking errors of estimating x_t for various situations of the reference robot

Robot Number	Fixed Ref. Robot	Mobile Ref. Robot	Ref. Robot With Controller
1	0.010591	0.02129	0.004189
2	62.54229	57.06329	0.013196
3	0.023677	0.034152	0.018124

Table. 12. Tracking errors of estimating y_t for various situations of the reference robot

6. Conclusion

This paper dealt with the problem of multi-robot tracking taken into account as one of the most important topics in robotics. The JPDAF algorithm was presented for tracking multiple moving objects in a real environment. Then, extending the aforementioned algorithm to robotics application was discussed. To enhance the quality of tracking, different motion models were introduced along with a simple near constant velocity model. Proposing a new approach for observer trajectory planning was the key part of this paper where it was shown that because of some problems such as increasing the variance of the additive noise by increasing the distance between targets and the reference robot, the tracking performance may be corrupted. Therefore, a fuzzy controller was proposed to find an optimal trajectory for the reference robot so that the effect of the additive noise is minimized. Simulation results presented in the paper confirmed the efficiency of the proposed fuzzy control approach in enhancing the quality of tracking.

7. References

Fox, D; Thrun. S; Dellaert. F & Bugard, W (2000). *Particle Filters For Mobile Robot Localization*, Sequential Monte Carlo in Practice, Springer, Verlag.

Kalman, R. E. & Bucy R. S. (1961). *New Results in Linear Filtering and Prediction*, Trans. American Society of Mechanical Engineers, Series D, Journal of Basic Engineering, Vol. 83D, pp. 95-108.

Siegwart. R. & Nourbakhsh. I. R. (2004). *Introduction to Autonomous Mobile Robots,* MIT Press.

Howard. A. (2005). *Multi-robot Simultaneous Localization and Mapping Using Particle Filters,* Robotics: Science and Systems I, pp. 201-208.

Anderson. B. D. O. & Moore. B. J. (1979). *Optimal Filtering,* Englewood Cliffs: Prentice-Hall.

Gordon. N. J. ; Salmon. D. J. & Smith. A. F. M. (1993). *A Novel Approach For Nonlinear/non-Gaussian Bayesian State Estimation*, IEE Proceedings on Radar and Signal Processing, 140, 107-113.

Doucet. A; Freitas. N. de. & Gordon. N. J. (2001). *Sequential Monte Carlo Methods in Practice,* Springer- Verlag.

Ristic. B; Arulampalam. S. & Gordon. N. J. (2004). *Beyond the Kalman Filter*, Artech House.

Ng. W; Li. J; Godsill. S. & Vermaak. J. (2004). *A Hybrid Approach For Online Joint Detection And Tracking For Multiple Targets*, In the Proceedings of IEEE Aerospace Conference.

Gustafsson. F; Gunnarsson. F; Bergman. N; Forssell. U; Jansson. J; Karlsson. R. & Nordlund. P. J. (2002). *Particle Filters For Positioning, Navigation, and Tracking,* IEEE Transactions on Signal Processing, Vol. 50, No. 2.

Freitas. N. de. (1999). *Bayesian Methods For Training Neural Networks,* PhD Thesis, Trinity College, The University of Cambridge.

Andrieu. C; Doucet. A; Singh. S. S. & Tadic. V. (2004). *Particle Methods For Change Detection, System Identification and Control,* Proceedings of IEEE, Vol. 92. No. 3.

Sarkka. S; Vehtari. A. & Lampininen. J. (2005). *Rao-Blackwellized Particle Filter for Multiple Target Tracking,* Information Fusion Journal, Vol. 8, Issue. 1, Pages 2-15.

Vermaak. J; Godsill. S. J. & Perez. P. (2005). *Monte Carlo Filtering for Multi- Target Tracking and Data Association*, IEEE Transactions on Aerospace and Electronic Systems, Vol. 41, No. 1, pp. 309–332.

Li. J; Ng. W. & Godsill. S. (2007). *Online Multiple Target Tracking and Sensor Registration Using Sequential Monte Carlo Methods*, In the Proceedings of IEEE Aerospace Conference.

Fortmann. T. E; Bar-Shalom. Y. & Scheffe. M. (1983). *Sonar Tracking of Multiple Targets Using Joint Probabilistic Data Association*, IEEE Journal of Oceanic Engineering, Vol. 8, pp. 173-184.

Schulz. D; Burgard. W; Fox. D. & Cremers. A. B. (2003). *People Tracking with a Mobile Robot Using Sample-based Joint Probabilistic Data Association Filters*, International Journal of Robotics Research (IJRR), 22(2).

Oh. S; Russell. S. & Sastry. S. (2004). *Markov Chain Monte Carlo Data Association for Multiple-Target Tracking,* In Proc. of the IEEE International Conference on Decision and Control (CDC), Paradise Island, Bahamas.

Singh. S. S; Kantas. N; Vo. B. N; Doucetc. A. & Evans. R. J. (2007). *Simulation-based Optimal Sensor Scheduling with Application to Observer Trajectory Planning,* Automatica, 43, 817 – 830.

Del Moral. P; Doucet. A & Jasra. A (2006). *Sequential Monte Carlo Samplers,* J. R. Statistics, Soc. B, Vol. 68, part 3, pp. 411-436.

Ikoma. N; Ichimura. N; Higuchi. T. & Maeda. H. (2001). *Particle Filter Based Method for Maneuvering Target Tracking,* IEEE International Workshop on Intelligent Signal Processing, Budapest, Hungary, pp.3-8,.

Pitre. R. R; Jilkov. V. P. & Li. X. R. (2005). *A comparative study of multiple-model algorithms for maneuvering target tracking*, Proc. 2005 SPIE Conf. Signal Processing, Sensor Fusion, and Target Recognition XIV, Orlando, FL.

Wang. X. L. (1997). *A Course in Fuzzy systems And Control,* Prentice-Hall, Inc, International Edition.

Symbiotic Evolution of Rule Based Classifiers

Ramin Halavati[1] and Saeed Bagheri Shouraki[2]
[1]*Iranian Academic Centre for Education, Culture, & Research*
[2]*Sharif University of Technology*
Iran

1. Introduction

Genetic Algorithm is a widely used approach in predictive data mining where data mining output can be represented by If-Then rules and discovering the best rules is done by a genetic algorithm. The main motivation for using genetic algorithms in discovery of high-level prediction rules is that they perform a global search in the problem space and cope better with attribute interaction in compare with greedy rule induction algorithms often used in data mining (Freitas, 2001) and therefore, one can see the following papers for a wide variety of representation techniques and evolution approaches in this field: (Teng et al, 2004), (Hasanzadeh et al, 2004), (Chen & Linkens, 2004), & (Cordon et al, 1998) for evolution of weighted fuzzy rule base with simple linear genetic representation; (Golez & Dasgupta, 2002) for rule base evolution with binary tree representation; (Mendes et al, 2001) for a co-evolutionary approach which evolves fuzzy rules in one process and fuzzy membership functions in another process; (Ishibuchi & Yamamoto, 2004), (de la Iglesia et al, 2003), & (Lopes et al, 1999) use multi objective optimization approaches for rule base evolution; (Ishibuchi & Yamamoto, 2002) & (Tsang et al, 2005) for two stage evolution in which one stage generates candidate rules and the other stage selects a combination of them as a final rule base; (Riquelme et al, 2003) for hierarchical representation; and some other variations in (Zhu & Guan, 2004), (Goplan et al, 2006), (Gundo et al, 2004), & (Eggermont et al, 2003).
There are two basic strategies for rule base evolution task and many hybrid methods that combine the good features of these two methods. These basic approaches are Michigan approach exemplified by Holland's classifier system (Holland, 1986), and the Pittsburgh approach exemplified by Smith's LS-1 system (Smith, 1983). In this chapter, we will first study these two schools with more details in section 2 and show why there is a need for a third school, then introduce natural process of symbiogenesis in section 3 and symbiotic evolution as a novel solution for this approach in section 4. Then section 5 will present the experimental and comparison results, followed by the summary and concluding remarks in section 6.

2. Michigan and Pittsburgh schools for rule-based classifier evolution

There are two basic strategies for rule base evolution task and many hybrid methods that combine the good features of these two methods. These basic approaches are Michigan approach, introduced by John Holland (Holland, 1986), and the Pittsburgh approach, popularized by Ken De Jong and Steve Smith (Smith, 1983).

In Pittsburgh approach, a number of if-then rules are coded as a string and handled as an individual. The performance of each rule-set (i.e., each individual) is used as its fitness value. Thus the genetic search for finding rule-sets with high fitness values is equivalent to the search for rule-based systems with high performance. Hence, the optimization of rule-based systems is directly handled by genetic algorithms that try to maximize the fitness function. Some good rule-sets in a current population are inherited to the next population with no modification as elite individuals. The performance of each rule is not explicitly evaluated in Pittsburgh approach. Thus even if good rules exist in the current population, they are not always used for generating new rule-sets. Especially when good rules are included in poor rule-sets, they easily disappear during the generation update. Since a population consists of a number of rule-sets, long computation time and large memory storage are required in Pittsburgh approach (Ishibuchi et al, 1999). Interested reader can see (De Jong et al, 1993), (Janikow, 1993), (Sen et al, 1997), & (Smith 1983) as good examples of this approach.

On the other hand, in Michigan approach where a single if-then rule is coded as a string and handled as an individual, the performance of each rule is used as its fitness value. That is, the performance of rule-sets (the entire population of current rules) is not utilized in the genetic search for finding rule-based systems with high performance. Thus the optimization of rule-based systems is indirectly performed by searching for good if-then rules. Performance of the current rule-set is not explicitly evaluated in the genetic search of the Michigan approach. Thus a good rule-set can be destroyed by the generation update (i.e. the performance of the current population can be decreased). Since a population includes only a single rule-set, computation time and memory storage in Michigan approach are much smaller than those in Pittsburgh approach where a population consists of a number of rule-sets. In Michigan approach, good if-then rules in the current population (i.e., in the current rule-set) are inherited with no modification to the next population. The generation update in Michigan approach can be viewed as a partial change of the current population where bad rules are replaced with newly generated rules. Thus once good if-then rules are found, they are not likely to disappear. (Ishibuchi et al, 1999). To see some good examples, one can check (Holland, 1986) and (Wilson, 1987).

There are three main viewpoints from which Pittsburgh and Michigan approaches can be compared: First, Pittsburgh approach seems to be better suited at batch-mode learning (when all training instances are available before learning is initiate) and for static domains, and Michigan approach is more flexible to handle incremental-mode learning (training instances arrive over time) and dynamically changing domains (Corcoran & Sen, 1994).

Second, considering that many classifier systems need to cover a complex state space in a small group of cooperative rules, one will see that this is in contrast to the nature of Michigan approach in which the rules are intrinsically competitive and the Pittsburgh approach is more suited to the provision of cooperation. This is because the lack of competition between individual classifiers in the Pittsburgh method allows the algorithm to find novel cooperative solutions that the population-level GA can maintain and proliferate. Therefore, Pittsburgh approach is usually the method of choice to apply to problems that require the development of cooperative populations (Barry et al, 2004).

The third item is very similar to the second: As evolving rules of a Michigan process are rivals and the general fitness value of the population has no effect in evolution, two problems occur: First, we usually need strategies for detection and prevention of redundant concept descriptions among population members (Liu & Kwok, 2000); Second, as a side effect of the first problem, a portion of training examples may be left unclassified and although the evolution would be at a stable position, there would be no rule for classification of this portion.

The fact that Pittsburgh is more powerful or easier to use for evolution of rule-sets in environments with complex concepts, where there is an urge for evolution of cooperative rules, makes it more attractive for most practical problems. However, the Pittsburgh approach presents its own limitation as well: In particular, because the evolution operates at a rule-set level, GA receives only high-level feedback from the fitness function and therefore cannot evaluate the role of individual rules in the success of a rule-set; hence, it requires a large additional effort to generate optimal populations. This increased effort in addition to the increased computational resource required to operate at the population level can present new challenges when devising efficient implementations for a Pittsburgh classifier evolution (Barry et al, 2004). This problem is a very important and known general problem of traditional genetic algorithms, called the linkage problem (Watson & Pollack, 2000).

Linkage problem has two parts: The first problem is called the problem of garbage or hitch-hiker genes (Forrest & Mitchel, 1993). In traditional GA, each chromosome may have a combination of good and bad genes which affect the total fitness value of the chromosome together. The effect of this problem in rule base evolution task is that a rule-set may have some rules with very good classification accuracy and some rules that have no positive effect or even have negative effects on the classification task. As evaluation is only done at rule-set level, selection or removal of all rules inside a rule-set is done together and there is no distinction between rules that have positive or negative effect on the classification. These bad rules (genes) inside a chromosome are called garbage genes or hitch-hiker genes because they gain their chance of survival by sticking to good genes as parastis.

The second part of Linkage problem is related to the recombination operator of genetic algorithms. During the process of this operator, some parts of the two parent chromosomes are extracted and merged with each other to create an offspring. Selection of appropriate parts from either of the parents has a great effect on the performance of the entire process, but there are many problem in which there is no way to identify the good sub-chromosomes. Here in rule-set evolution, one of the interesting features of the Pittsburgh approach is the evolution of cooperative rules inside a rule-set, but using a crossover operator separates the rules of one rule-set from each other and then blindly combines them with some from another rule-set, with no guarantee that these parts match each other or be able to help each other in a common classification task.

Many different recombination operators or alternative evolution strategies are introduced to cope with linkage problem in GA, such as designing more sophisticated recombination operators for simple genetic algorithms such as the ones with more number of cut points, random cut point positioning, uniform crossover, linear combination of genes, etc., see (Mitchell, 1999) for an extensive list; use of chromosome reordering operators and repositioning of genes inside the chromosome on the fly such as Inversion operator (Bagley, 1967) and Linkage Learning Genetic Algorithm (Harrik, 1997); and algorithms based on partially specified chromosomes such as Messy Genetic Algorithms (mGA) (Deb, 1991), (Goldberg et al, 1989), Cooperative Co-Evolutionary Algorithms (CCEA) (Potter & De Jong, 1994), Symbiotic Evolutionary Adaptation Model (SEAM) (Watson & Pollack, 2000), and Incremental Commitment Genetic Algorithm (ICGA) (Watson & Pollack, 1999).

As far as the authors know, except CCEA approach which is partially used in some tasks and some special purpose recombination operators, none of the other above approaches have been used in rule base evolution and the major efforts in rule-based classifier evolution to cope with linkage problem have been in hybridizations of Michigan and Pittsburgh approaches to add the positive features of both methods together, such as (Ishibuchi et al, 1999) & (Tan et al, 2003). Not commenting on the applicability or generality of these hybrid approaches, we present a novel pure approach based on Symbiotic evolution instead of Genetic evolution to solve this problem in the rest of this chapter. It must be emphasized that we introduce this algorithm as a basic approach comparable to pure Pittsburgh and therefore, it is not compared with hybrid approaches or extensions of other algorithms as all such hybridizations or extensions can be studied for this algorithm as well. Section 3 will represent the natural bases of this approach and section 4 will have all the details.

3. The natural process of symbiogenesis

The natural process of symbiogenesis (Merezhkovsky, 1909) is the creation of new species from the genetic integration of organisms, called symbionts. Symbiogenesis has enabled some of the major transitions in evolution (Maynard Smith & Szathmary, 1995), including the origin of eukaryotes which include all plants and animals. This kind of genetic integration is quite different from the transfer of genetic information in sexual reproduction. Sexual recombination occurs between similar organisms (i.e. of the same species) and involves the exchange of parts of the genome in a mutually exclusive manner; that is, every gene acquired from one parent is a gene that cannot be acquired from the other parent. In contrast, symbiotic combination may also occur between genetically unrelated organisms (i.e. different species) and involve the integration of whole genomes. The resultant composite may have all the genes from one symbiont and at the same time acquire any number of genes from the other symbiont (Watson & Pollack, 2000).
Based on this idea, symbiotic combination operator is introduced (Watson & Pollack, 1999) & (Watson & Pollack, 2000) as an alternative for sexual recombination operator. Symbiotic combination operator is applied to partially specified chromosomes, i.e., chromosomes which have some positions with unspecified values. This operator takes two partially specified chromosomes and makes an offspring with the aggregation of their characteristics of both of them; see Fig. 1 as an example. Therefore, in contrast to the standard crossover operator that receives two fully specified chromosomes and creates one/two individuals that have received each of their genes from either parents, this operator runs over two/more partially specified representations and creates an offspring with can have even all genes of both/all parents.

```
Chromosome A:          1--1---0
Chromosome B:          --00-111
A + B:                 1-01-110
```

Fig 1. An example of symbiotic combination. Chromosomes A and B, each, have some unspecified locations, shown with '-' mark. Their combination has specified values for all locations that are specified in at least one of the donors. If there would be a conflict between the specified values, like the last gene of the above chromosomes, all conflicts are resolved in favor of one donor, here A.

This can be very beneficent for evolution of rule based classifiers in Pittsburg approach because each individual (chromosome) is a complete classifier. Therefore, its rules are a collection and they have proved to work good together. Separating them for a recombination and combining some parts of them with parts of another classifier may disrupt the functionality of both classifiers. On the other hand, adding them, assuming that each of them is a relatively good classifier, just adds up their classification powers.

4. Symbiotic evolutionary algorithm

The basic idea of Symbiotic Evolutionary Algorithm (SEA) is to replace the crossover operator of Pittsburgh genetic algorithm (PGA) with symbiotic combination operator. To do so, the evolution starts with rule-sets (individuals) which have just one rule (gene). During the process, similar to traditional PGA, evaluation and selection is done at rule-set level. Mutation operator is also quite similar to conventional PGA, but instead of crossover operator, sometimes two rule-sets combine using symbiotic combination and create an individual with more rules. If this combination shows a higher accuracy in compare to its parents, the parents are removed from the population and the offspring remains, otherwise, the offspring is neglected.
In this section, we first present our rule-set model which is used both in SEA and the PGA that is used in next section for comparisons. Then will move on the details of the Symbiotic Evolutionary Algorithm.

4.1 Rule-set model and fitness values

To emphasize on the algorithm, we have a chosen a very simplistic representation for our fuzzy rules, taken from (Hasanzadeh et al, 2004), but we still insist that SEA is not dependent to this model or the fuzzy nature of the rules. In this model, each rule is a horn clause, with If-part consisting of fuzzy membership functions for different features of the problem data base, and Then-part stating the class to which this rule belongs. A rule-set is composed of one or more rules, with each rule having a weight value stating its role in final decision. To classify an input by a rule-set, each of the rules computes the degree of similarity between the input and its own If-part and based on that, it states a degree of belief to its Then-part. Then, a weighted sum of the degree of beliefs for each class is computed and the class which gets the highest value is chosen. Fig. 2 specifies the structure of the rule-set.

```
<RULE-SET>  → a set of <RULE>s
<RULE>      → <WEIGHT> + a set of <CONDITION>s + <RESULT>
<WEIGHT>    → a real value
<RESULT>    → a Class Name
<CONDITION> → a <FEATURE> [IS / ISNOT] a <MEMBERSHIP FUCNTION>
<FEATURE>   → one of the features of dataset.
<MEMBERSHIP FUCNTION> →   one of the possible fuzzy values for the
                          respective feature.
```

Fig .2. Formal structure of the rule set (chromosome)

The fitness of each rule-set is defined as the accuracy of the rule-set in classification of all training data. Accuracy is a measure combining the classification soundness with 99.9

percent effect and the simplicity of the rules with 0.1 percent effect. The simplicity measure is used to break the tie between two rule-sets with different complexities and similar classification rate, in favor of the simpler rule-set. Simplicity of a rule-set is computed as stated in equation 1.

$$\text{Simplicity} = \frac{1 + \text{Number of rules with just one condition}}{\text{Total number of conditions in all rules}} \tag{1}$$

4.2 The algorithm

The Symbiotic Evolutionary Algorithm starts by generating a population of random rule-sets, each having just one rule. In each iteration of the algorithm, a set of rule-sets with high fitness values are selected using a tournament selection algorithm; they will be called the *Selected Set* hence forth. After selection, each of these individuals undergoes a mutation and all mutants are added to the population. The mutation operator is presented in Figure 3.

```
Function Name: MUTATATION

Summary:     Takes a rule set and mutates it.

Input: Rule Set R.
     Assume $R = \{R_1,R_2,...,R_n\}$ and each $R_i$ as
     $[Weight+ (F_1,C_1,MF_1) \wedge (F_2,C_2,MF_3) \wedge ... \wedge (F_m,C_m,MF_m),\ Class]$ where each
     $F_j$ is feature, $O_j$ is a condition(Is/Is Not), and $MF_j$ is a
     membership function from the domain of $F_j$.

Function Detail:
  1. Randomly choose $R_i$ from $R_1$ to $R_n$. Set m to the number of rules in
     $R_i$.
  2. Randomly select one of the next steps and apply it on $R_i$:
        a. Increase or decrease Weight.
        b. Choose j from 1..m, remove $(F_j,C_j,MF_j)$ from $R_i$.
        c. Randomly generate a new (F,C,MF) and concatanate it to $R_i$.
        d. Choose j from 1..m, reverse $C_j$ so that Is becomes IsNot,
           and IsNot becomes Is.
        e. Choose j from 1..m, change $MF_j$ to a random new membership
           function from the domain of $F_j$.
  3. Return.
```

Fig. 3. Pseudo Code of the Mutation Operator

After mutation, instead of the conventional cross over operator, symbiotic combination operator is applied over the selected set. The operator takes two members of the selected rule-sets and merges them, so that the combination includes all rules of both sets. If the child strictly outperforms both of its parents, the combination will be added to the population; otherwise, it will be discarded. To control the growth speed of the number of rules in each rule-set, there is another control mechanism that limits the size of the largest rule-set that can be added to the population at a time. This parameter, which will be called *SizeLimit*, is 1 at the beginning and limits the size of rule-sets to just one rule. During the process, *SizeLimit* is increased with a selected strategy, and allows emergence of rule-sets with more number of rules. In all of our implementations, we have set the control strategy to a simple linear function of iterations count, but one may use a more complicated function, if it looks fit.

Fig. 4 presents the pseudo code of Symbiotic Evolutionary Algorithm.

```
Algorith Name: Symbiotic Evolutionary Algorithm

Summery:     Takes a database of training examples and generates a
             rule-set to classify them, using symbiotic combination
             operator and Mutation function.

Parameters:  SR:  Selection Rate
             TS:  Tournament Size
             RC:  Random Rule Creation Rate
             MP:  Maximum Population

Algorithm Detail:
  1.  INITIALIZATION:
          a.  Generate a population of random rule-sets, each having
              just one rule.
  2.  PROCESS CONTROL:
          a.  Update SizeLimit (Initialized to 1).
          b.  If Best generated rule set is satisfactory, return it
              and exit.
  3.  SELECTION PHASE:
          a.  Create an empty set called SelectedSet.
          b.  For SR x PopulationSize times,
                  i.  Randomly pick TS rule-sets from the pool, add the
                      best one to SelectedSet.
  4.  MUTATION PHASE:
          a.  For each memer of SelectedSet such as rs,
                  i.  Create a mutated copy of rs using Mutation
                      function, call it rs'.
                 ii.  Add rs' to the pool.
  5.  SYMBIOTIC COMBINATION PHASE:
          a.  For each two members of the SelectedSet such as rs1 and
              rs2,
                  i.  Create the symbiotic combination of rs1 and rs2
                      and call it rs3.
                 ii.  If SizeOf(rs3) < SizeLimit and fitness value of
                      rs3 exceeds that of rs1 and rs2,
                          Add rs3 to the pool.
  6.  DIVERSITY MAINTANANCE:
          a.  Create RC random new rule-sets and add them to the pool.
  7.  POPULATION CONTROL:
          a.  While PopulationSize is above MP limit, randomly select
              and remove some random rule-sets from the poool.
  8.  Goto Step 2.
```

Fig. 4. Pseudo Code of Symbiotic Evolutionary Algorithm

5. Experimental results

5.1 Test conditions

There are too many classification approaches and also many extensions to basic genetic based classifiers. But as we are introducing SEA as a basic new approach, we have just compared it with pure Pittsburgh GA in detail. More comparisons can be done in future works.

To compare the performance of SEA algorithm with Pittsburgh GA, we used six frequently used benchmarks: The first one is a 10% selection of KDDCUP99 dataset (MIT Lincoln Labs, 2007) and others are selected from University of California Irvine, Machine Learning Repository (Blake & Merz, 1998); these datasets are gathered from real experiments, so they can show efficiency of the algorithm in some real circumstances. Credit Approval (CRX), Glass Identification (Glass), Iris Plant (Iris), 1984 United States Congressional Voting Records Database (Vote), and Wine Recognition (Wine) datasets are selected as the most frequently used datasets so as to compare the results to some other related works. The extensive information about these datasets is mentioned in Table 1. Although KDDCUP99 data set has many classes of intrusion types, we consider their classes as Normal and Attack cases, similar to (Esposito et al, 2005), (Toosi & Kahani 2007), and (Mill & Inoue, 2004). General specifications of benchmarks are expressed in Table 1. The GA algorithm is implemented as described in (Hasanzadeh et al, 2004) with exactly the same parameters (expressed in Table 2).

Likewise (Hasanzadeh et al, 2004) & (Hasanzadeh & Bagheri, 2003), Fuzzy C-Mean clustering (Zimmermann, 1996) was used to define the fuzzy membership function for continuous attributes, and fuzzy singletons were defined for none-parametric attributes. The number of fuzzy sets for KDD99 features is 5 and for other problems, 3 fuzzy sets are created. The exact parameters of SEA algorithm are presented in Table 3.

The tests are done four-fold (Blake & Merz, 1998), i.e. the data was randomly divided into 4 sets and in each trial, one set was taken as test set, and the other 3 were used as training set. Each test is repeated for 20 times, and the average, minimum and maximum classification rates for training and tests results are depicted in subsection 5.2 tables. The stopping criterion of each run is an unchanging best fitness value during 5000 fitness function calls.

Also, the average number of fitness function calls to reach the highest classification accuracy and the average ratio between time and fitness function calls for each benchmark/algorithm is reported in subsection 5.3 as a measure of algorithms speed.

Dataset	Features count	Numeric Features	Nominal Features	Classes	Instances
KDD99	41	34	7	2	494021
CRX	15	6	9	2	690
Glass	10	9	1	6	214
Iris	4	4	0	3	150
Vote	16	0	16	2	435
Wine	13	13	0	3	178

Table 1. Datasets Specification

Parameter	Value
Maximum Population	200
Mutation Rate	0.7
Elitism Rate	0.2
Tournament Size	4

Table 2. Pittsburgh GA Parameters, as in (Hasanzdeh, 2003)

Parameter	Value
Population Size	1000
Selection Rate	6
Tournament Size	8
Random Creation Rate	4

Table 3. SEA Parameters

5.2 Accuracy comparison results

Tables 4 and 5 represent the classification rates of Pittsburgh Genetic Algorithm (PGA) and SEA training and test data, respectively. As presented there, SEA has found better rule-sets in compare with PGA in all cases on training sets and 4 of 5 on test sets.

Data Sets	PGA			SEA			Average SEA to PGA Improvement[1]
	Min	Max	Average	Min	Max	Average	
CRX	87.433	88.937	88.07	85.199	90.042	88.85	6.54 %
Glass	63.921	72.023	69.42	66.923	74.812	71.43	6.57 %
Iris	98.139	99.082	98.63	97.237	99.91	99.35	52.55 %
Vote	96.528	98.003	97.32	96.474	97.976	97.56	8.96 %
Wine	96.189	99.156	97.68	99.153	99.910	99.44	75.86 %
KDD99	87.433	88.937	88.07	85.199	90.042	88.85	6.54 %

Table 4. Average Classification Rate of PGA and SEA, Different Data Sets, on Training Data

Data Sets	PGA			SEA			Average SEA to PGA Improvement
	Min	Max	Average	Min	Max	Average	
CRX	83.746	87.654	85.27	84.888	86.476	85.58	2.1 %
Glass	63.377	71.370	68.62	67.878	74.008	70.68	6.56 %
Iris	91.85	99.923	94.95	91.805	99.909	95.57	12.28 %
Vote	91.789	98.661	95.31	92.611	97.972	95.04	-5.76 %
Wine	86.293	99.902	92.9	90.821	97.683	94.59	23.8 %
KDD99	84.263	87.654	94.36	85.156	85.16	99.31	87.77 %

Table 5. Average Classification Rate of PGA and SEA, Different Data Sets, on Test Data

Table 6 presents the best classification results of some other approaches ((Gomez et al, 2002), (Mendes et al, 2001), (Liu & Kwok, 2000), & (Rouwhorst & Engelbrecht, 2000)) which are reimplemented and tested by (Hasanzadeh, 2003) with similar settings as ours. As stated there, in cases that we had sufficient comparison data, SEA is better than other algorithms in all data sets.

Also Table 7 presents some other results from other papers that have used almost similar test specifications with that of ours. It must be emphasized that the test condition of these results does not fully comply that of ours, in some cases not exactly specified and in other slightly easier or harder. As depicted there, SEA is among the top 2 best results for all benchmarks.

[1] (SEA – PGA) / (100 – PGA)

Algorithm	CRX	Glass	Iris	Vote	Wine	KDD'99
Fuzzy Classifier with Expression Tree Representation (Gomez et al, 2002)			94.84	85.42	92.22	
Fuzzy Classifier with Co-Evolution (Mendes et al, 2001)	84.7		95.3			
Extended Genetic Rule Induction (Liu & Kwok, 2000)	77.39	72.43	95.3			
Evolution of Decision Trees (Rouwhorst & Engelbrecht, 2000)			94.1			
SEA	85.58	70.68	95.57	95.04	94.59	99.31

Table 6. Classification rate of some other algorithms with exactly similar settings in compare to SEA, from (Hasanzadeh, 2003).

Algorithm	CRX	Glass	Iris	Vote	Wine	KDD99
Fuzzy Kohonen Network (Lorenz et al, 1997)			91.33			
Fuzzy Classifier System (Lorenz et al, 1997)			96.00			
ID3 (Dong & Kothari, 2003)	81.16					
Naive Bayes (Dong & Kothari, 2003)	77.68					
Bayesian Network (Ezawa & Schuermann,1995)	86.5					
C 4.5 (Ezawa & Schuermann,1995)	85.5					
Discrimination Analysis (Ezawa & Schuermann,1995)	83.4					
Fuzzy Classifier System (Ishibuchi & Yamamoto, 2005)		68.22				
k-means (Guo et al, 2006)		63.08	92.67		68.54	
MLP Neural Network (Ueda, 2000)		70.3				
Hyper Sphere SVM (Liu et al, 2007)		62.15	95.68			
MLP Neural Network (Deodhare et al, 2007)					95.8	
Rule Extraction based on Grey Lattice Classification (Yamaguchi et al, 2005)					86.7	
Tree Support Vector Machine (Mill & Iune, 2004)						70.75
Array Support Vector Machine (Milll & Iune, 2004)						91.30
Fuzzy Rule Base with Linear Tree Genetic Representation (Dasgupta & Gonzalez, 2001)			94.5	94.7	93.9	
Average of above approaches	*82.84*	*65.93*	*94.03*	*94.7*	*86.23*	*81.02*
Best of above approaches	*86.5*	*70.3*	*96.00*	*94.7*	*95.8*	*91.30*
SEA	85.58	70.68	95.57	95.04	94.59	99.31

Table 7. Average Classification Rate of some other algorithms with almost similar test settings in compare to SEA.

5.2 Speed comparison results

Figures 5-10 depict the best fitness values over time for SEA and GA on the six stated datasets, averaged in all runs. As it is presented in the diagrams, SEA has found a better solution much faster than GA in all cases. Table 8 summarizes these results, and presents the average time taken to find the best result by each algorithm on each benchmark. As stated there, SEA has reached its best result notably faster than GA in all cases.

Also Figure 11 depicts the relation between number of fitness function calls and time for the two algorithms. Four curves show GA and SEA algorithms for CRX and Iris datasets which are, respectively, the largest and the smallest UCI ML Repository datasets used in this paper. The curves are almost linear with a slight trend toward taking more time for each fitness function call while the algorithms are proceeding. Thus, the progress of elite fitness can be considered through either time or fitness function calls in diagrams 5 to 10. Number of fitness function calls can be considered as a rough measure of the speed complexity of the algorithm as it removes the effects of programming details on algorithm speed.

Dataset	SEA	PGA	SEA to PGA Improvement
CRX	357	4650	92.32 %
Glass	164	280	41.42 %
Iris	40	633	93.68 %
Vote	89	1490	94.02 %
Wine	98	1710	94.26 %
KDD99	7012	54306	87.08 %

Table 8. Average time taken by SEA and Pittsburgh GA to find the best classifier on different benchmarks, in seconds.

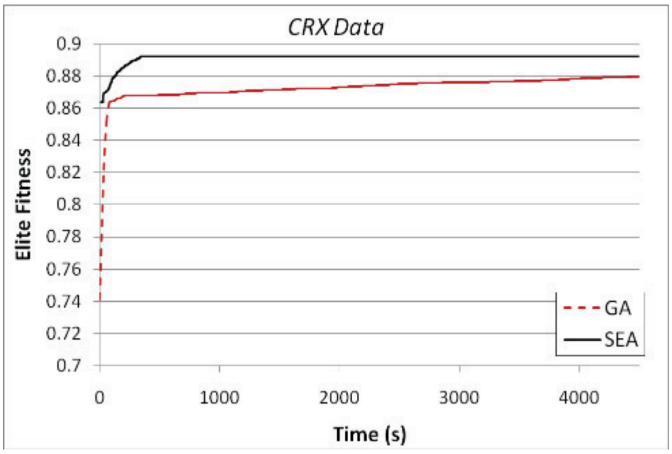


Fig. 5. CRX benchmark, average fitness of best rule set found by Pittsburgh GA and SEA over time.

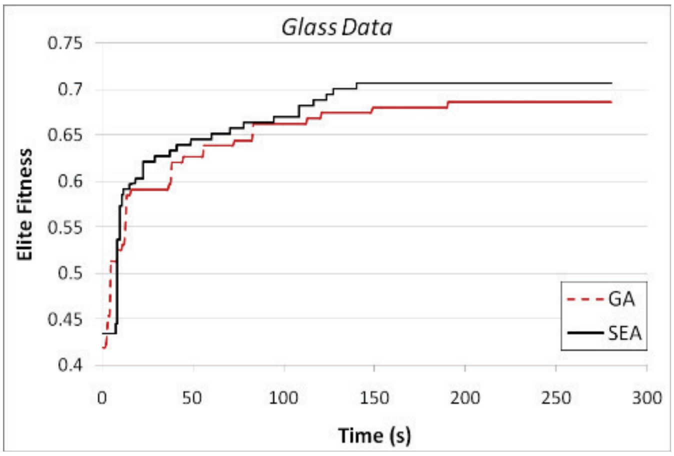


Fig. 6. Glass benchmark, average fitness of best rule set found by GA and SEA over time.

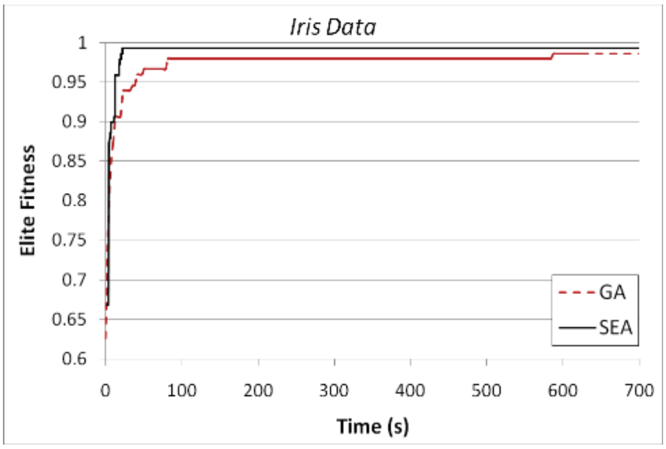


Fig. 7. Iris benchmark, fitness of best rule set found by Pittsburgh GA and SEA over time.

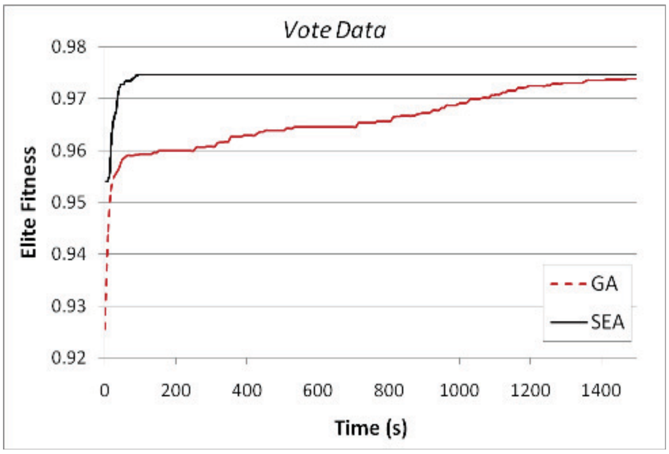


Fig. 8. Vote benchmark, fitness of best rule set found by Pittsburgh GA and SEA over time.

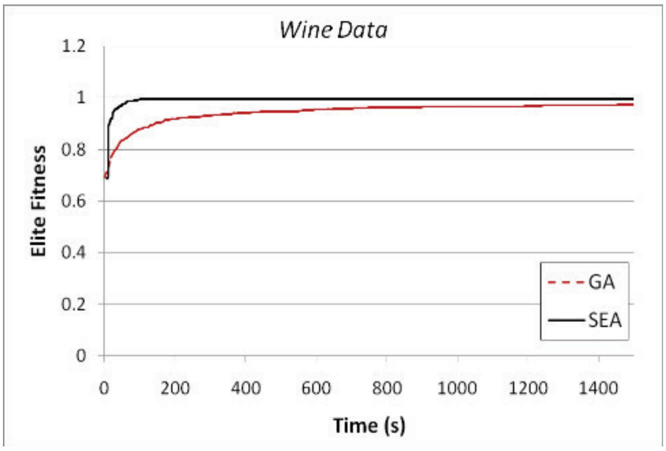


Fig. 9. Wine benchmark, fitness of best rule set found by Pittsburgh GA and SEA over time.

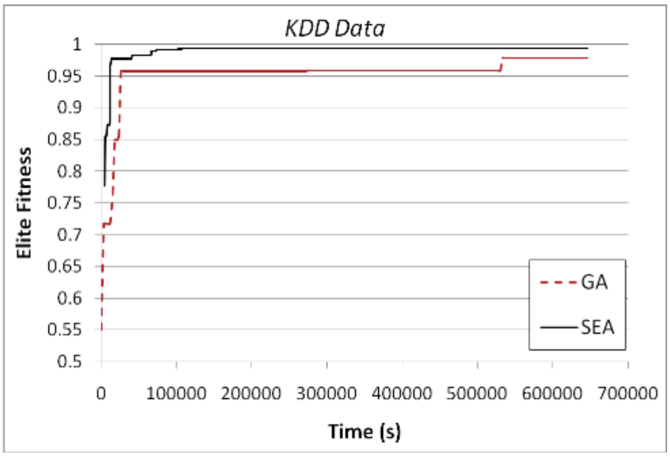


Fig. 10. KDD benchmark, fitness of best rule set found by Pittsburgh GA and SEA over time.

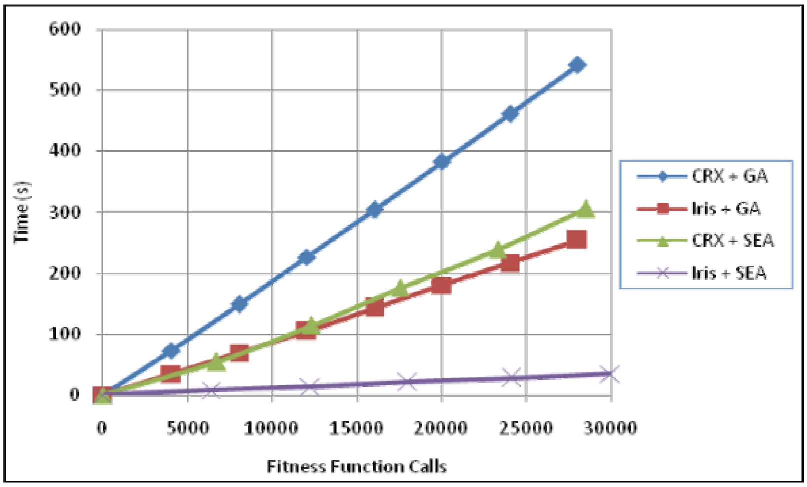


Fig 11. Time versus Fitness Function Calls, GA and SEA algorithms, CRX and Iris datasets.

6. Summary and concluding remarks

While the suitability of evolutionary approaches for generation of rule based classifier systems is shown in many different contributions, the structure and elements of this process are an important issue in design of a system that works efficiently. As stated in section 2, Michigan algorithm is faster and requires less memory in compare to Pittsburgh algorithm, but it has two very important problems that makes Pittsburgh the favorite one in many cases: First, the cooperation of single rules that are all evolved for better classification, regardless of other rule's behavior, will not necessarily result in a general good classifier; Second, some parts of the problem space might be neglected.

Pittsburgh evolutionary algorithm also has three problems that must be dealt with during an efficient implementation: First, how to recombine two rule-sets? While traditional sexual recombination operators splits the two parents and merges their parts, how should one know which rules of either rule-set (parents) must be extracted to be recombined to make a good combination. Second, what to do with the parasite rules? And the third question is how many rules must a rule-set have to get a small, but accurate classifier?

SEA algorithm uses symbiotic combination operator instead of common sexual recombination operator of GA, and provides a solution for the three above questions; it creates an offspring from two parents by combining all of their rules (genes), and adds the offspring to the gene pool only if it outperforms both its parents. Using this strategy, SEA avoids grouping separate rules before it makes sure that the group works better than the isolated ones, so it avoids garbage rules. It doesn't break any generated rule-set; therefore, it doesn't require a method to identify good working sub sets of two rule-sets. Also, as it grows the rule-sets only if growing results in better performance, the designer does not need to make a decision about chromosome sizes in advance.

Experimental results clearly comply with this hypothesis where SEA had 6 to 75 percent classification error reduction on training data in compare with Pittsburg GA and 2 to 87 percent on test data, except in one case which resulted in 6 percent more classification error. Moreover, this significant better accuracy was reached by 41 to 92 percent less computation time, in similar operating conditions.

As SEA is introduced as a basic algorithm to resolve the problems of Pittsburgh algorithm, we have just compared it in details with Pittsburgh GA, but some accuracy comparisons with algorithms from other families were also presented in section 6 and 7. Although some of these comparisons are not very fair as they were taken from different sources with slightly different test conditions, SEA presented very good comparison results to all of them as well.

Table 9 presents a features summary of SEA, Michigan, & Pittsburgh algorithms. As it is noted there, SEA stands between Michigan and Pittsburgh approaches from many viewpoints, collecting the positive points of both of them. SEA starts with light weight single rule individuals, as in Michigan, and gradually evolves them towards complete rule-set individuals, as in Pittsburgh. Due its growing size of individuals, it stands between Michigan and Pittsburgh in speed and memory complexity measures. Similar to Pittsburgh, it allows cooperation inside rule-sets but unlike Pittsburgh and similar to Michigan, this does not result in parasite rules, keeping rule-sets neat and accurate. Inheritance is done both on rule level and rule-set level as there is no distinction between rule and rule-sets. As the fitness of a rule-set is defined over all of its rules, a single rule that correctly covers a small uncovered portion of training samples can increase the credit of a rule-set and therefore is accepted and added to the rule-set, so, unlike Michigan approach there is no need to set specific credit to less frequently used training samples. And at last, in contrast to

Pittsburgh that blindly recombines two rule-sets, SEA combines two rule-sets only if this increases the overall recognition performance.
As next stages of this task, we can recommend an extra function that recognises rules that have redundant effects after symbiotic combinations. Also more specific representations and local optimization of rule-sets may result in better classification rates.

	Michigan	Pittsburgh	SEA
Individual	A single rule	A rule-set	Starts with single rules and reaches rule-sets
Selection and Evaluation	On each rule	On each rule-set	On each rule-sets
Rules Cooperation	None, Rules are rivals	Cooperative inside rule-sets, rival among rule-sets	Cooperative inside rule-sets, rival among rule-sets
Garbage Rules	Not Existing	Severely Existing	Not Existing
Computation Time	Least	Most	Between others
Memory Size	Least	Most	Between Others
Rule Optimization	Direct	Indirect	Both direct and indirect
Inheritance	Good Rules	Good Rule-Sets	Both good rules and rule-sets
Requires class credit assignments	Yes	No	No
Requires rule-set size specification	Yes	Yes / Controlled by a score function	No, controlled by accuracy.
Rule-Set recombination	None	Yes, but may result in lower accuracy	Yes, always results in higher accuracy.

Table 9. Feature Comparison of Michigan, Pittsburgh and SEA.

7. Acknowledgements

Authors wish to thank Mr. Pooya Esfandiar and Ms. Sima Lotfi for their help during implementation and testing of this task and Ms. Maryam Hasanzadeh for sharing the details of her implementation, test, and data sets.

8. References

Bagley, J.D. (1967). *The Behaviour of Adaptive Systems Which Employ Genetic and Correlation Algorithms,* PhD Dissertation, University of Michigan.

Barry, A., Holmes, J., & Llor, X. (2004). Data Mining using Learning Classifier Systems, In: *Applications of Learning Classifier Systems, Bull, L. (Ed.), 15-67, Springer,* ISBN:3540211098.

Blake C.L., Merz C.J. (1998). UCI Repository of machine learning databases, Irvine, CA: University of California, Department of Information and Computer Science. http://www.ics.uci.edu/~mlearn.

Chen, M.Y., Linkens, D.A., (2004). *Rule-base self-generation and simplification for data-driven fuzzy models*, Fuzzy Sets and Systems, Volume 142, Issue 2, 1 March 2004, Pages 243-265.

Chi-Ho Tsang; Sam Kwong; Hanli Wang, (2005). Anomaly intrusion detection using multi-objective genetic fuzzy system and agent-based evolutionary computation framework, in Proceedings of Fifth IEEE International Conference on Data Mining.

Corcoran, A.L., & Sen, S. (1994). *Using real-valued genetic algorithms to evolve rule sets for classification*. In Proceedings of the IEEE Conference on Evolutionary Computation, pages 120--124, 1994.

Cordon O., del Jesus M.J., Herrera F., (1998). *Genetic learning of fuzzy rule-based classification systems cooperating with fuzzy reasoning methods*, International Journal of Intelligent Systems 13 (10–11) 1025–1053.

Dasgupta, D. & Gonzalez, F., (2001). Evolving Complex Fuzzy Classifier Rules Using a Linear Tree Genetic Representation, In L. Spector, D. Whitley, D. Goldberg, E. Cantu-Paz, I. Parmee, and H. Beyer, editors, Proc. of the Int. Conf. on Genetic and Evolutionary Computation (GECCO-2001), pages 299-305. Morgan- Kaufmann, San Francisco, CA.

De Jong, K.A. , Spears, W., & Gordon, D.F. (1993). Using genetic algorithms for concept learning. *Machine Learning*, 13(2-3):155–188.

de la Iglesia B., Philpott M.S., Bagnall A.J., Rayward-Smith V.J., (2003), *Data Mining Rules Using Multi-Objective Evolutionary Algorithms*, in Proceedings of IEEE Congress on Evolutionary Computations, Vol. 3, pp 1552-1559.

Deb, K. (1991). *Binary and floating point function optimization using messy genetic algorithms* (IlliGAL Report No. 91004). Urbana: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Deodhare, D., Murty, M. N., and Vidyasagar, M., (2007). *A Unified Approach to Encoding and Classification using Bimodal Projection-based Features*, in Proceedings of the International Conference on Computing: Theory and Applications (ICCTA'07) pp. 348-354.

Dong, M. & Kothari, R., (2003). *Feature subset selection using a new definition of classifiability*, Pattern Recognition Letters 24 (2003) 1215–1225.

Eggermont J., Kok J.N., Koster W.A., (2003) *Genetic Programming for Data Classification: Refining the Search Space*, in Proceedings of the Fifteenth Belgium/Netherlands Conference on Artificial Intelligence, pp 123-130.

Esposito M., Mazzariello C., Oliviero F., Romano S. P., Sansone C., (2005). *Evaluating Pattern Recognition Techniques in Intrusion Detection Systems*, in Proceedings of the 7th International Workshop on Pattern Recognition in Information Systems (PRIS 2005) - 24-25 May 2005, Miami, FL (USA) pp. 144-153.

Ezawa, K. J., & Schuermann, T., (1995). *A Bayesian network Based Learning System: Architecture and Performance Comparison with Other Models*, in Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty, pp 197 – 206.

Forrest S., Mitchell M. (1993). *Relative Building-block fitness and the Building-block Hypothesis*. In Whitley, D, ed. FOGA 2, Morgan Kaufmann, San Mateo, CA.

Freitas, A., *A survey of evolutionary algorithms for data mining and knowledge discovery*. In Advances in Evolutionary Computation. Springer- Verlag, 2001.

Goldberg, D.E.; Korb, B. & Deb, K. (1989) Messy Genetic Algorithms: Motivation, analysis, and first results. *Computer Systems*, 3, 5, 493-530.

Gomez, J., Dasgupta, D., (2002) *Evolving Fuzzy Classifiers for Intrusion Detection*, in Proceedings of the 2002 IEEE Workshop on Information Assurance.

Gomez, J., Gonzalez, F., Dasgupta, D., (2002). *Complete Expression Trees for Evolving Fuzzy Classifier Systems with Genetic Algorithms*, in Proceedings of the Evolutionary Computation Conference GECCO'02, 2002.

Gopalan J., Alhajj R., Barker J., (2006). *Discovering Accurate and Interesting Classification Rules Using Genetic Algorithm*, in Proceedings of the 2006 International Conference on Data Mining, pp. 389-395. June 26-29, 2006.

Gundo K.K., Alatas B., Karci A., (2004). *Mining Classification Rules by Using Genetic Algorithms with Non-random Initial Population and Uniform Operator*, Turkish Journal of Electrical Engineering and Computer Science, Vol.12, No. 1, 2004.

Guo, H.X., Zhu, K.J., Gao, S.W., & Liu, T., (2006). *An Improved Genetic k-means Algorithm for Optimal Clustering*, in Proceedings of Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06) pp. 793-797.

Harik, G.R. (1997). *Learning Gene Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithm*, PhD Dissertation, University of Illinois at Urbana-Champaign, Urbana, Illinois.

Hasanzade M., (2003). *Fuzzy Intrusion Detection*, MS. Dissertation, Computer Engineering Department, Sharif University of Technology, Tehran, Iran, 2003.

Hasanzade, M., Bagheri, S., Lucas, C., (2004). *Discovering Fuzzy Classifiers by Genetic Algorithms*, in Proceedings of 4th international ICSC Symposium on Engineering of Intelligent Systems (EIS2004), 2004, Island of Madeira, Portugal.

Holland, John H. (1986). Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In R. Michalski, J. Carbonell, and T. Mitchell (Eds.), Machine Learning: An Artificial Intelligence Approach (Vol. 2). Morgan Kaufmann Publishers, Los Altos, CA.

Ishibuchi H., Nakashima T., and Murata (1999). T., *A hybrid fuzzy genetics-based machine learning algorithm: Hybridization of Michigan approach and Pittsburgh approach,* in proceedings of IEEE, fuzzy IEEE.

Ishibuchi H., Yamamoto T., (2002). *Fuzzy rule selection by data mining criteria and genetic algorithms,* in Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2002), pp. 399-406, New York, July 9-13.

Ishibuchi H., Yamamoto T., (2004). *Fuzzy Rule Selection by Multi-Objective Genetic Local Search Algorithms and Rule Evaluation Measures in Data Mining,* Fuzzy Sets and Systems, Vol. 141, no. 1, pp. 59-88, January 2004.

Ishibuchi, H. &, Yamamoto, T., (2004). *Rule Weight Specification in Fuzzy Rule-Based Classification Systems* , IEEE Transactions on Fuzzy Systems, vol. 13, no. 4, August 2005.

Janikow, C.Z. (1993) A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 13(2-3):180–228.

Liu J.J., & Kwok J.T. (2000). *An Extended Genetic Rule Induction Algorithm*, in Proceedings of IEEE Congress on Evolutionary Computation (CEC-2000). La Jolla, CA, USA. July 2000.

Liu, S., Liu, Y., Wang, B., and Feng, X., (2007). *An Improved Hyper-sphere Support Vector Machine*, in Proceedings of the Third International Conference on Natural Computation (ICNC 2007) pp. 497-500.

Lopes C., Pacheco M, Vellasco M, Passos E, (1999). *Rule-Evolver: An Evolutionary Approach For Data Mining*, in Proceedings of the 7th International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing, RSFDGrC'99, pp 458-462.

Lorenz, A., Blum, M., Ermert, H., & Senge, T., (1997). *Comparison of Different Neuro-Fuzzy Classification Systems for the Detection of Prostate Cancer in Ultrasonic Images*, in Proceedings of the IEEE Ultrasonics Symposium, 1997, Volume: 2, pp 1201-1204.

Maynard Smith, J., Szathmary, E. *The Major Transitions in Evolution*, WH Freeman: Oxford UK, 1995.

Merezhkovsky, K. S. (1909), *The Theory of Two Plasms as the Basis of Symbiogenesis, a New Study or the Origins of Organisms*. In Proceedings of the Studies of the Imperial Kazan University, Publishing Office of the Imperial University, (In Russian).

Mendes, R., R., F., Voznika, F., de B., Freitas, A., A., Nievola, J. C., (2001). *Discovering Fuzzy Classification Rules with Genetic Programming and Co-Evolution*, In Principles of Data

Mining and Knowledge Discovery (Proc. 5th European Conference PKDD 2001) – Lecture Notes in Artificial Intelligence, Springer-Verlag.

Mill J., Inoue A. (2004). *Support Vector Classifiers and Network Intrusion Detection*, in Proceedings of IEEE Conference on Fuzzy Systems 2004, Vol 1. pp 407-410.

MIT Lincoln Labs, (2007). KDD CUP 99 DARPA Intrusion Detection Dataset, http://kdd.ics.uci.edu/databases/kddcup99.

Mitchell, M. (1999). *An Introduction to Genetic Algorithms*, MIT Press, 0−262−13316−4, London, England.

Potter, M.A., De Jong, K.A. (1994). A Cooperative Coevolutionary Approach to Function Optimization. *In: Parallel Problem Solving from Nature (PPSN III)*, Y. Davidor, H.-P. Schwefel and R. Manner (Eds.). Berlin: Springer-Verlag, 249-257.

Riquelme J.S., Toro J.C., Aguilar-Ruiz M., (2003), *Evolutionary Learning of Hierarchical Decision Rules*, in IEEE Transactions on Systems, Man, and Cybernetics, Vol. 33, Issue 2, pp 324-334.

Rouwhorst, S.E., Engelbrecht A.P., (2000). *Searching the Forest: Using Decision Tree as Building Blocks for Evolutionary Search in Classification*. In Proceedings of IEEE Congress on Evolutionary Computation (CEC-2000), 633-638. La Jolla, CA, USA. July 2000.

Sen, S., Knight, L., & Legg, K. (1997). Prototype based supervised concept learning using genetic algorithms. In D. Dasgupta and Z. Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*, pages 223–239. Springer.

Smith, S. F. (1980). *A Learning System Based on Genetic Adaptive Algorithms*, PhD Thesis, University of Pittsburgh.

Smith, S. F. (1983). Flexible Learning of Problem Solving Heuristics Through Adaptive Search, Proc. 8th IJCAI, August 1983.

Tan K.C., Yu Q., Heng C.M., Lee T.H., (2003). *Evolutionary computing for knowledge discovery in medical diagnosis*, Artificial Intelligence in Medicine 27, pp.129-154.

Teng M., Xiong F., Wang R., Wu Z., *Using genetic algorithm for weighted fuzzy rule-based system*, in Proceedings of Fifth World Congress on Intelligent Control and Automation, 2004.

Toosi A.N., Kahani M. (2007). *A New Approach to Intrusion Detection Based on an Evolutionary Soft Computing Model Using Neuro-Fuzzy Classifiers*, Computer Communications, Vol 30, 2201-2212.

Ueda, N., (2000). *Optimal Linear Combination of Neural Networks for Improving Classification Performance,* IEEE Transactions on Pattern Analysis and Machine Learning, vol. 22, no. 2, February 2000.

Watson, R.A. & Pollack, J.B. (1999), Incremental Commitment in Genetic Algorithms, *Proceedings of GECCO'99.*, Morgan Kaufmann, 710-717.

Watson, R.A., Pollack, J.B. (2000). *Symbiotic Combination as an Alternative to Sexual Recombination in Genetic Algorithms*, in Proceedings of Parallel Problem Solving from Nature (PPSN VI).

Wilson, S. (1987). Classifier systems and the Animat problem. *Machine Learning*, 2:199–228.

Yamaguchi, D., Li, G.D., Mizutani, K., and Akabane, T., (2005). *Decision Rule Extraction and Reduction Based on Grey Lattice Classification*, in Proceedings of the Fourth International Conference on Machine Learning and Applications, 2005, 15-17 Dec. 2005.

Zimmermann, H., J., (1995). *Fuzzy Set Theory and Its Application*, Kluwer Academic Publishers.

Zhu F., Guan S.U., (2004). *Ordered Incremental Training with Genetic Algorithms*, International Journal of Intelligent Systems, Volume 19, Issue 12 , pp 1239-1256.

A Multiagent Method to Design Open Embedded Complex Systems

Jamont Jean-Paul and Occello Michel
University of Grenoble, LCIS/INPG-UPMF Lab
France

1. Introduction

Open physical complex systems involve multiple interconnected software and hardwar entities which enable logical/physical interactions between them and their shared environment. They rise to many hierarchical level which exhibit common behaviours. These entities have their own goals but participate to the accomplishment of the global system.
There are different classes of open physical complex systems like control systems processing systems, communication systems and interactive systems. Because these systems take over new wireless technologies, they are more and more distributed, decentralized and often not completely described.
Through the use of multiagent system to model Open physical complex systems (OCPS) two types of requirements emerge: requirements in methods and in specific system architectures. Concerning the specific methods, our contribution is the DIAMOND method (Decentralized Iterative Approach for Multiagent Open Networks Design (Jamont & Occello, 2007)). Concerning the requirements in architecture, our contribution is the MWAC model (Multi-Wireless-Agent Communication) based on our previous work on wireless sensor networks (Jamont & Occello , 2006).
In this chapter, we focus on specificities of the methodological requirements. We try to answer to some questions asked by this type of applications in lifecycle terms, about the design step and the formalism.
A method consists in concepts, in an approach and in tools. So, in a first section, we focus on the main concept of our works: the multiagent paradigm. In a second part, we present the approach of the DIAMOND method. The third part describes the different steps and activities of our method. Before concluding, we propose a discussion of the method in comparison to other multiagent methods.

2. Multiagent systems

An agent is a software entity evolving in an environment that it can perceive and in which it acts. It is endowed with autonomous behaviours and has objectives. Autonomy is the main concept in the agent issue: it is the ability of agents to control their actions and their internal states. The autonomy of agents implies no centralized control (Wooldridge, 1999).
A multiagent system is a set of agents situated in a common environment, which interact and attempt to reach a set of goals. Through these interactions a global behaviour, more

intelligent than the sum of the local intelligence of multiagent system components, can emerge.

The emergence paradigm deals with the unprogrammed and irreversible sudden appearance of phenomena in a system confirming that "the whole is more than the sum of each part". It is one of the expressions of collective intelligence (Deguet et al., 2006).

The emergence process is a way to obtain dynamic results from cooperation that cannot be predicted in a deterministic way. There are three types of emerging features (Marcenac, 1996): emergence of structures at the origin of the self-organization process, behaviour and emergence of properties.

It is difficult to qualify the emergent characteristics of a phenomenon. Some fundamental elements have been settled by S. Forrest (Forrest, 1991),(Muller; 2004) proposes an interesting specialization in the multiagent context that has been recently discussed and completed in (Dessales & Phan, 2005).

It asserts that a phenomenon is emergent if:

- there is a set of agents interacting via an environment, whose state and dynamics cannot be expressed in terms of the emerging phenomenon to produce in a vocabulary or a theory D,
- the dynamic of the interacting agents produces a global phenomenon such as, for example, an execution trace or an invariant,
- the global phenomenon is observable either by the agent (strong sense) or by an external observer (weak sense) in different terms from the subjacent dynamics i.e. another vocabulary or another theory D '.

To give a system of agents a particular global functionality, the traditional method consists in carrying out a functional decomposition of the problem into a set of primitives which will be embodied by the agents. The alternative suggested by L. Steels (Steels, 1990) aims at making this functionality emerges from the interactions between the agents. The advantage of the "emergent functionality" approach is first of all a reinforcement of the robustness of the system becoming less sensitive to the changes of the environment.

The adaptation of the whole multiagent system is generally obtained through emergence. It exist a lot of multiagent methods. We give here some references to these different works and the result of an analysis of these methods through many criteria.

3. Approach

The lifecycle of traditional methods applied to design hardware/software hybrid systems (see fig.1) starts with a requirements analysis followed by a portioning step. During this partitioning step, the designer chooses the system parts which must become either hardware or software parts %: the requirements analysis which is derived in a hardware one and a software one. At this stage, the two different parts are designed in parallel. At the end of the lifecycle, the two parts are integrated into a whole operational system. Through this integration step (and the following tests) some problems can emerge. These problems can question the software design, the hardware design or the both. Furthermore, it can be necessary to modify the whole result of the partitioning!

This type of lifecycle doesn't allow to take into account some late modification of requirements and is thus not well adapted to OPCS which cannot, by definition, be completely a priori specified.

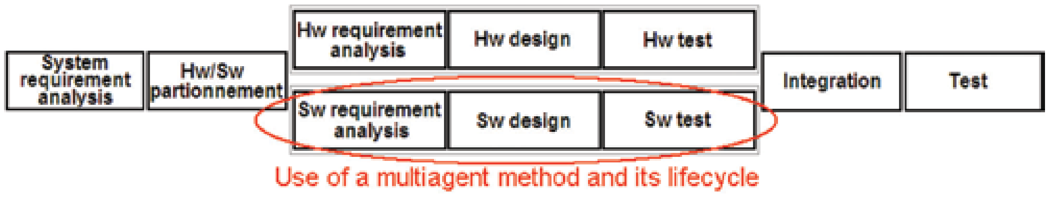


Fig. 1. Lifecycle of a traditional multiagent method

A few works deal with embedded multiagent systems, but new applications are strongl concerned by this domain (Pervasive computing (Carabelea et al., 2003), Ambiant computing (Maña & Rudolf, 2007)) and industrial applications of MAS (Parunak, 2000)). Even if we are at the beginning of the expansion of embedded multiagent systems, we are sure that embedded MAS methods will be the continuation of traditional embedded system design lifecycle (see fig 1). Multiagent approaches focus on software parts and forget the hardware aspects. Hardware aspects are generally taken into account only during the deployment step (Cossentino03 et al.), and are limited to the choice of the platform where the agents must be deployed.
We can thus say that the hardware/software hybrid systems design is very partially covered by MAS methods. An alternative to this type of lifecycle is the codesign approach. A codesign method unifies the development of both hardware and software parts by the use of a unified formalism. The partitioning step is pushed back at the end of the life cycle. We can thus settle at this point of our study that the choice of a specific lifecycle model which supports a codesign approach is required.
Because of the complex features of our system, the lifecycle model must enable late modification of specifications. Furthermore, it is necessary to come back on previous design steps (refinement) and to explore the solution space of the hardware/software compromise. The design process must accept genericity (incremental criteria are in favour of the genericity). Finally, we must identify and keep a trace of all the parameters of the different retained solutions. The evaluation of different lifecycle models in respect with these previous criteria leads to adopt a spiral lifecycle (Boehm, 1988).
The lifecycle of traditional method applied to design an hardware/software hybrid system (see fig.1) begin by a requirement analysis followed by, very early, by a portioning step. During this partitioning step, the designer chooses the system part which must become hardware part or software part: the requirement analysis which is declined in a hardware one and a software one. After this step, these two different parts are designed in parallel. At the end of lifecycle, these two parts are integrated to become operational system. Through this integration step (and the following test) some problem can emerge. Theses problems can call into question the software design, the hardware design or the twice. More deeply, it can be necessary to modify the result of the partitioning!
The evaluation of the different lifecycle models in respect with these previous criteria carries out the spiral lifecycle (Boehm, 1988) as the best choice in our context.
The DIAMOND method is built to design physical multiagent systems. Four main stages, distributed on a spiral cycle (see fig.2), may be distinguished within our physical multiagent design approach. The definition of requirements defines what the user requirements are and characterizes the global functionalities. The second stage is a multiagent-oriented analysis which consists in decomposing a problem in a multiagent solution. The third stage of our

method starts with a generic design which aims to build the multiagent system, once one knows what agents have to do without distinguishing hardware/software parts. Finally, the implementation stage consists in partitioning the system in a hardware part and a software part to produce the code and the hardware synthesis.

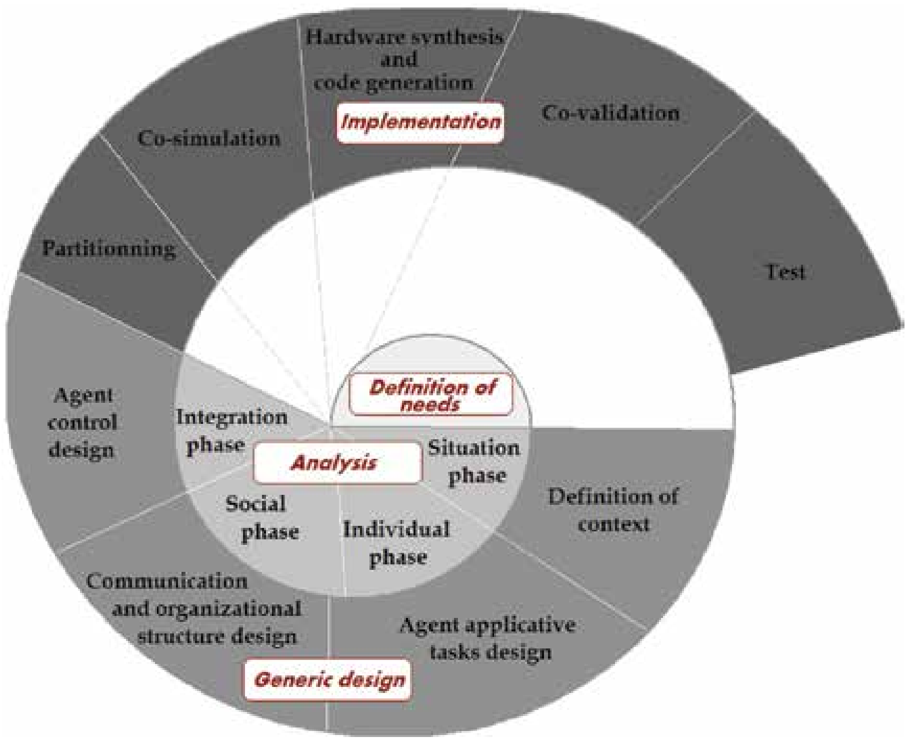


Fig. 2. Lifecycle of a traditional multiagent method

4. The DIAMOND method

4.1 Case study

To illustrate the various phases and activities of our method, we will use the robocup case study. To make the illustration easily understandable, we will adopt a simplified definition of requirements.

The experimental conditions are inspired by (Huang et al., 2001). Robots evolve on a football field (see fig. 3). A video recorder system makes possible to know the position of each robot as well as of the ball. These positions are periodically broadcasted to all robots. If the ball goes out of the limits of the field, a robot of the non faulty team recovers the ball and plays (the order is given by the referee). If a robot has no more battery or is dysfunctioning, the match is stopped (the order is given by the referee for human safety reasons) and the robot is withdrawn from the field: all robots must be then motionless. At the beginning of a match the robots must be located in their camp and the referee decides to give the guardian role to one robot of each team. So, the game is open and the team, which scores the higher number of goals in 90 minutes, wins.

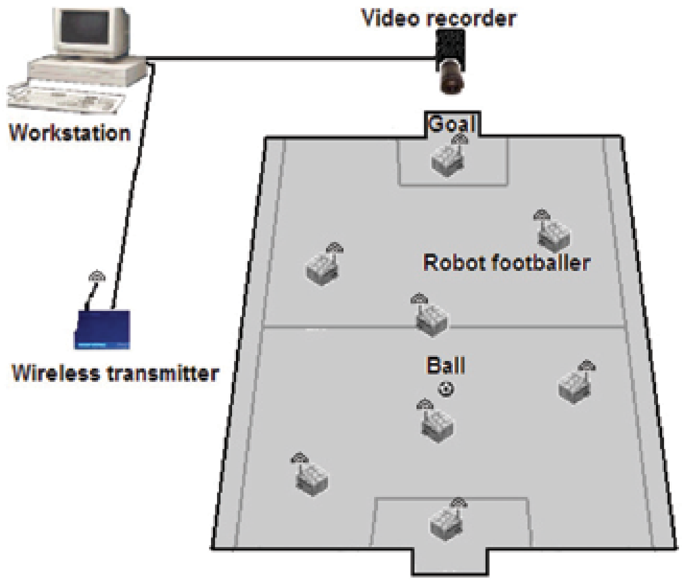


Fig. 3. Our case study

4.2 Definition of requirements

This preliminary stage begins by analyzing the physical context of the system (identifying workflow, main tasks, etc...). Then, we study the different actors and their participative user cases (using UML use case diagrams), the services requirements (using UML sequence diagram) of these actors. The UML sequence diagram can include physical interaction.

The second step consists in an original step: the study of the running mode and stop mode.

This activity is very significant because it enables to structure the global running of the system. It is generally wishable that the system works in autonomy. But working with physical systems requires to identify many others possible behaviours: how must the system be before to stop it (robot in safety area...)? What must the system states be when it goes under maintenance? How must the system components be calibrated? What must the state of all the components be when an emergency stop occurs? Even if the problem is solved with a decentralized intelligence, this organization of these modes is easily understandable by the clients and the users. More of that, even if the system is approached with a decentralized intelligence, the system must respect laws and norms. They are very strong because the human safety can easily be altered.

This activity puts forward a restricted running of the system. It allows to specify the first elements necessary for a minimal fault-tolerance. Moreover, it enables to identify cooperative (or not) situations and to define recognition states in order to analyze, for example, the self-organizational process of an application. This activity allows to take into account the safety of the physical integrity of the users possibly plunged in the physical system.

We have defined 15 different modes regrouped in three families. The *stop modes* are relate to the different procedures for stopping the system. Moreover it allows to define the associate recognition states. The *running modes* focus on the definition of the recognition states of normal running, test procedures etc. The *failing operations modes* focus on the security

procedures (for example to allow a human maintenance team to work in the system) or to specify rules for restricted running etc.
Application to our case study. We find the following actors. The *referee (logical actor)* manages the match parameters: choose a goalkeeper and a camp for each team, verifies that robots respect the rules. It authorizes the human to withdraw a robot when all robots are motionless.
The *manager (physical actor)* withdraws robots when a problem occurs. The *ball (physical actor)* moves under the robot actions. The opposing team (*physical/logical actor*) shares the field with the studied one.
The *camera system* broadcasts the coordinates of each robot and of the ball.
There are two user cases. The *configuration* expresses that the referee chooses a field and a goalkeeper for each team. This user case triggers another one: the *games* opens the game (see fig.4).

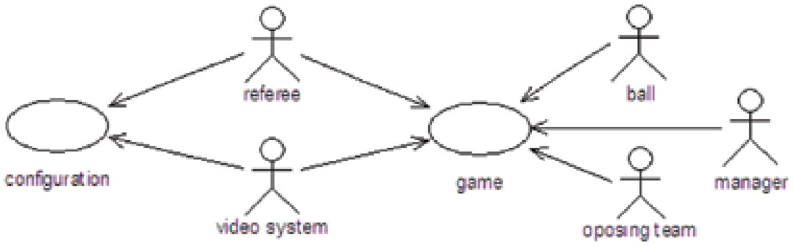


Fig. 4. Our case study

For our application, the identified modes are:
- Stops modes: Two modes of stops must be characterized: other modes are not exploited.
  - Idle: In a idle mode, the robots must be motionless.
  - Stops requested on normal mode: when a robot dysfunction occurs, the referee can decide to freeze the game.
- Running modes:
  - Normal mode: in this mode all the robots must answer to requests of the referee, there is no emergency stop.
  - Mode of preparation: during the phase of preparation, robots are positioned on the ground. Robots should neither then move nor use their actuators. This mode ends when the parameters setting period starts.
  - Mode of test: this mode will be used to calibrate the shooting power.
- Failure modes: only the management of the emergency stop is relevant in our application.
  - Mode of stop aiming to ensure the safety: If an emergency stop is activated, robots do not have any more the right to move or use effectors.

In this application, where the life period is short, importance of the other modes is not relevant.

4.3 Multiagent oriented analysis
The multiagent stage is handled in a concurrent manner at two different levels. At the society level, the multiagent system is considered as a whole. At the individual level, the system's agents are built. This integrated multiagent design procedure encompasses five main phases discussed in the following.

Situation phase. The situation phase defines the overall settings, i.e., the environment, the agents, their roles and their contexts. This stems from the analysis stage. We first examine the environment boundaries, identify passive and active components and we proceed to the agentification of the problem.
We insist here on some elements of reflexion about the characteristics of the environment (Russel & Norvig, 1995),(Wooldridge, 2000). We must identify here what is relevant to take into account from the environment, in the resulting application.
It's, first of all, necessary to determine the environment *accessibility* degree i.e. what can be perceived from it. We will deduce from these characteristics which are the primitives of perception needed by agents. Measurements make possible to measure parameters which enable to recognize the state of the environment. They thus will condition the decisional aspect of the agent. The environment can be qualified of *determinist* if it is predictable by an agent, starting from the environment current state and from the agent actions. The physical environment is seldom deterministic. Examining allowed actions can influence the agent effectors definition. The environment is *episodic* if its next state does not depend on the actions carried out by the agents. Some parts of a physical environment are generally episodically. This characteristic has a direct influence on agent goals which aim to monitor the environment. Real environment is almost always *dynamic* but the designer is the single one able to appreciate the level of dynamicity of the part of the environment in which he is interested. This dynamicity parameter has an impact on the agent architecture. Physical environments may require reactive or hydride architectures. The environment is *discrete* if the number of possible actions and states reached by the environment are finite. This criterion is left to the designer appreciation according to the application it considers. A real environment is almost always continuous.
It is then necessary to identify the active and passive entities which will compose the system. These entities can be in interaction or be presented more simply as the constraints which modulate these interactions. It is necessary to specify the role of each entity in the system. This phase allows to identify the main entities that will be used and will become agents.
Application to our case study. The environment is not accessible. Each robot can know its geographical position, the position of the ball and of the other robots. Dimensions of the ground are known and the field of each team is communicated at the beginning of each part. The positions of each robot can be memorized at different dates to estimate displacements, directions of the robots and their trajectories. The trajectory of the ball obeys to physical laws. Agents can estimate this trajectory and act on it. Environment is rather not determinist. Even if agents cooperate and there is no dysfunction, an agent cannot know actions of other agents. However elements of the environment are not fully predictible like the trajectory of the ball. The possible actions on the environment are displacements (robots and ball). Environment is not episodical because we suppose that no intervention of the human is possible. The future evolutions depend only on the actions carried out by the robots. Environment is dynamic and continuous although the feasible actions are finite.
The active entities are the robot-players. The ball is a passive entity which obeys to agent actions (shootings) by a displacement according to the physical laws.
Individual phase. Decomposing the development process of an agent refers to the distinction made between the agent's external and internal aspects. The external aspect deals with the definition of the media linking the agent to the external world, i.e., what and how the agent can perceive, what it can communicate and according to which type of interactions, and how it can make use of them.

The agent's internal aspect consists in defining what is proper to the agent, i.e. what it can do (a list of actions) and what it knows (its representation of the agents, the environment, interaction and organization elements (Demazeau, 1995).
In most cases, the actions are carried out according to the available data about the agent's representation of the environment. Such a representation based on expressed needs has to be specified during specifications of actions. In order to guarantee that the data handled are real data, it is necessary to define the required perception capabilities. We have defined four types of actions. *Primitive actions* are tasks which are not physically decomposable. *Composed actions* are temporal ordered lists of primitives. *Situated actions* need to have a world representation to execute their tasks.
Application to our case study. The agent world representation consists in a collection of triplets (id,x,y) and in the field dimension. In our application, robot players are modelled by agents. Their individual capabilities can be specified using a tree to show the different action levels (fig. 5).

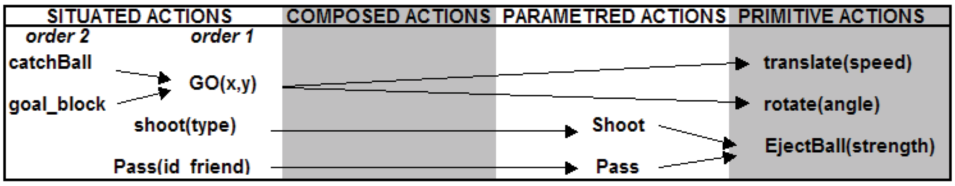


Fig. 5. Actions scheme

We specify the agent context with a context diagram (see fig 6).
After one iteration to take into account the society phase, individual behaviours are implemented using finite state machine. We can define an agent with the goalkeeper behaviour. Other agents can alternate two different behaviours (shooter or defender). For example, the goalkeeper behaviour defines that the agent must always be on a possible trajectory of shooting.

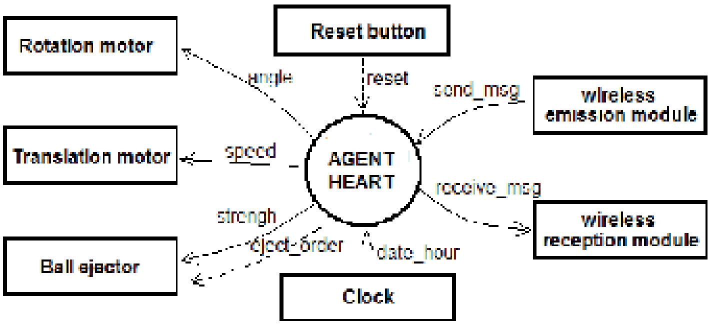


Fig. 6. Context diagram

Society phase. Interactions among agents are achieved via messages passing. Such exchange modes are formalized by means of interaction protocols. Although these interaction protocols are common to all the agents, they are rather external to them. Conflict resolution is efficiently handled by taking into account the relationships between the agents, that is, by building an explicit organizational structure. Such an organization is naturally modelled through subordination relations that express the priority of one agent on another.

Application to our case study.
Representation of others: The positions of other players can be known by the capture of information from the video system (WIFI module). Their directions can be estimated if agents can memorize the previous positions. Friend's intentions can be announced.
Interactions: between the agents they are carried out by exchange of messages. An agent must be able to communicate with its team to diffuse its intention. It can use a peer-to-peer communication to solve a conflict or to choose a trajectory with a friend.
Collaborative actions can be instantiated: a player can request the ball when it has an occasion for shooting. It can ask somebody to change position to attract an opponent elsewhere.
Organization: A TEAM according to the requirement is composed of a goalkeeper and three other agents which can be SHOOTER or DEFENDER.
Collective behaviour can be implemented by finite state machines.
Integration phase. We need to analyse the possible influences upon the previous levels. Those influences are integrated within the agents by means of their communication and perception assessment capabilities (given in each agent's model through guard and trigger rules). The decomposition masks the notion of agent's control, i.e., how it handles its focus of attention, its decisions, and how it links its actions. This dual aspect is based on the two previous one. Through the integration of social influences within the agents, one will endow the multiagent system with some dynamics. According to the social analysis we must give to the agent the possibility to interact in order to choose its role.
Application to our case study. We illustrate this phase with two examples.
Influence: If an agent wants to move to a given point, somebody (a friend or not) can be on its trajectory. Correction: If the agent on the trajectory is a friend, the agent owning the ball has the priority.
Influence: Two agents request the ball for shooting. Correction: Agents use an election protocol (they exchange an estimation of their success probabilities).

4.4 The generic design

This stage is based on component decomposition. We can define a component as an elementary object, which performs a specific function that allows developers to define reusable segments of code. It is designed in such a way to easily operate with other components to create an application. So, a component is a reusable program building block, which is an identifiable part of a larger program. Components can be combined with others to build more complex functions. This phase offers an efficient process leading to component decomposition by starting from the informal description of the multiagent system built during the previous stage.
The Problem Description Phase. This phase consists in identifying and delimiting the domain of the general problem, as well as identifying some specific aspects that should be taken into account. Although this phase is informal, it allows designers to clearly separate the various aspects embedded within the application. We must choose here the architecture of the different agents.
The agents are built following hybrid architectures, i.e. a composition of some pure types of architecture. Indeed, the agents will be of a cognitive type in case of a configuration alteration, it will be necessary for them to communicate and to manipulate their knowledge in order to have an efficient collaboration. On the other hand, in a normal mode use it will be necessary for them to be reactive using a stimuli/response paradigm to be most efficient.

Application to our case study At this level, the designer chooses technical solutions for each sensors/effectors. The context diagram (fig. 6) is detailed (see the table 1).
Using a hybrid architecture for the agents enables to combine the strong features of each of reactive and cognitive capabilities seen before. We use our ASTRO hybrid architecture (Occello et al., 1998), especially adapted to a real time context.

Information	Specification
Reset	Active on high logical level (1bit)
Angle	Relative angle in [- 180, +180] coded whole signed on 10 bits
Speed	Two speeds are possible. Entirety coded on 2 bits. (00: stop / 01: slow speed / 10: fast speed)
Strengh	Two levels of possible forces. Level coded on 1 bit (0: pass/1: shooting)
Eject_ball	Transition to high level
Date_heure	Number of milliseconds run out since the powering (32 bits)
Send_msg	Specific protocol bit field (sender 1octet, receiver 1byte, data_lenght 1octet, data 1-25octets)
Receive_msg	Specific protocol bit field (same than Send_msg)

Table 1. Details of the context diagram

Agent applicative tasks design phase. We must build the external shell of the agent i.e. elaborating the interface with the external world for each sensors and effectors. It is time, here, to choose technological solution for them and to complete the context diagram to specify all information about the signal. The next step is to design the internal shell of the agent. We begin by the elaborated actions according to the task tree.
It is necessary at this stage to arrange the components to build the application: the architecture of the agent will be used as a pattern, at a very high level, for the components decomposition.
The components have an external and an internal description. The internal description can be an assembly of components, or a formatted description of a decisional algorithm.

4.5 Implementation stage

Partitioning Phase. The main use of codesign techniques appears in the software/hardware partitioning of the components defined in the third level. Also it is essential to study the different partitioning criteria.
A first level relates to agent parts for which the partitioning question doesn't exist. Indeed some elements must be hardware as input/output peripherical such as for example the sensors and the actuators.
The second level relates to features for which there are several choices of implementation. We present below, those which can be considered to be relevant for the agents according to previous works we have made in this field (Occello et al., 1998),(Jamont et al., 2002),(Luo et al., 2007) and codesigns work like (Adams & Thomas, 1996):

- The *cost* is present at all the stages of a system design life cycle. On very small series, we must decrease, as much as possible, the price of the software/hardware development and the hardware material. In the case of great series, we must reduce manufacturing costs.
- The *performance* depends on the considered problem. A real-time application for which the robustness is a function of the occupation processor time is an example of system where this criterion is very important. A hardware partitioning is often privileged.
- The f*lexibility* plays in favour of the software. Software modifications have generally a less significant impact on the whole system than a hardware change. However, the flexibility of the EPLD (Electrical Programmable Logic Device) and other FPGA (Field Programmable Gate Array) increases quickly. For example, these architectures are reprogrammable in-situ : it is possible to modify their specifications without extracting them from the electronic chart.
- From their nature, software systems are fewer *faults tolerant* than hardware components like EPLD. Indeed, microcontrollers use memories, stack structures with possible overflow etc. The *internal fault tolerance* will be thus a criterion which will play in favour of a hardware partitioning.
- The *ergonomic constraints* gather all the system physical characteristics like weight, volume, power consumption, thermal release etc. Depending on the application, this criterion can be highly critical (case of the aeronautics embedded applications). One more time, the designer must appreciate correctly this criterion.
- The *algorithmic complexity* has a great importance for some applications. The software part will be more important if tasks are very complex. In fact, it is very difficult to make hardware synthesis of highly cognitive features.

Co-simulation and co-validation Phases. This activity allows to simulate the collaboration between software part, hardware part and their interface.

Implementation Phase. At this level, each component is completely specified with common graphic specification formalism for the hardware part and the software part. For each component, the designer has already selected if he wishes a hardware or a software implementation.

This level must ensure the automatic generation of the code for the components for which implementation software has been selected. The code is made in a portable language like Java or C++.

We use a Hardware Description Language which provides a formal or symbolic description of a component or of a hardware circuit and it interconnections. In our method the hardware components are specified in VHDL (Breuer et al. , 1999). The compilation of the code and the hardware synthesis of different specifications in VHDL are carried out like illustrated on figure 7.

Application to our case study. Today, the agents are embedded on autonomous processor cards. These cards are equipped with communication modules and with measuring modules to carry out agent tasks relative to the instrumentation. These cards supply a real time kernel. The KR-51(the kernel's name) allows multi-task software engineering for C515C microcontroller. We can produce one task for one capability. We can then quite easily implement the parallelism inherent to agents and satisfy the real-time constraints.

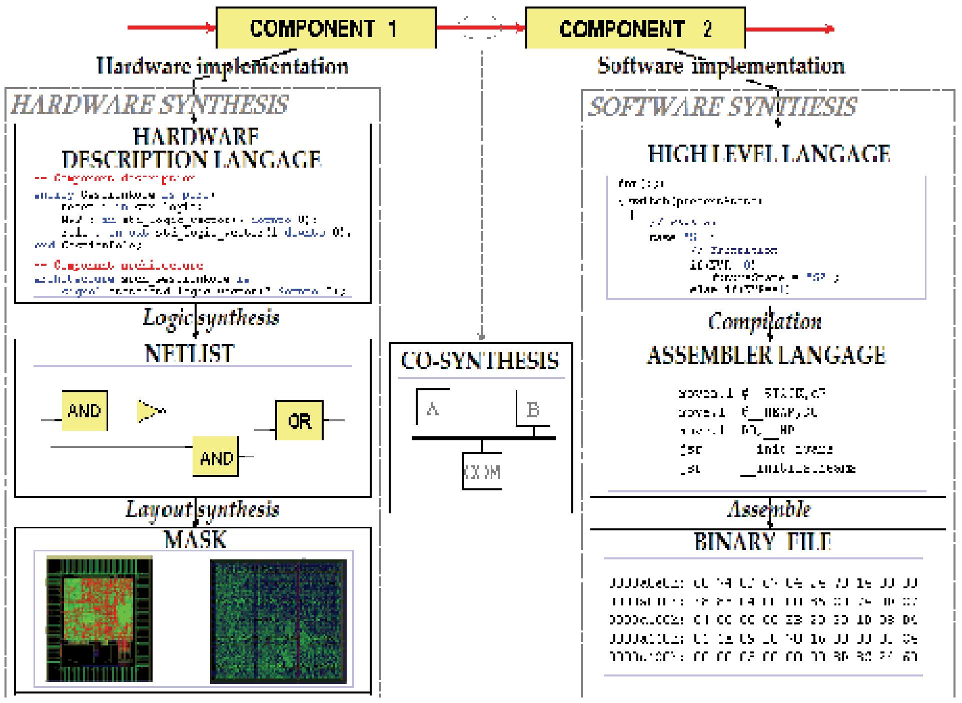


Fig. 7. Software component synthesis and hardware component synthesis

5. Discussion about the DIAMOND method

5.1 Lifecycle and phases

Most existing multiagent methods usually distinguish only analysis and design phases (Deloach et al., 2001). Very few methods deal with other phases. We can find for example a deployment phase in MASSIVE or Vowels. This deployment phase takes in our particular field a great importance since it includes the hardware/software partitioning. A last and major difference between DIAMOND and other multiagent approach is, as said previously, that DIAMOND unifies the development of the hardware part and the software part. In a traditional system design, the partitioning step stands at the beginning. In fact, a hardware requirement and a software requirement are created from the system requirements.
The software part of the system is built using a multiagent method and its associated lifecycle.
To cover the whole lifecycle, different formalisms are required to express different things at different levels (Herlea et al., 1999), for this reason we adopt a lifecycle using four stages mixing different expressions using more or less formal paradigms and languages (agents, components, Finite State Machines, Hardware Definition Languages). The most current lifecycle used in multiagent methods is the classical cascade lifecycle. Even if some works attempt to introduce iterative cycles as Cassiopeia (W) or Gaia, the proposal of a spiral lifecycle is very original.
In the definition of requirements phase, we introduce a study of the modes of running and stops to structure the global running of the system. In the generic design phase, the design allows an abstraction of the software design and the hardware design. We use components to build the agents as few multiagent methods introducing an actual componential

dimension (Lind, 2001),(Brazier et al., 2002). These components are used to simplify the work of the designer through visual programming, to manage the complexity through a functional decomposition, to increase the genericity through reusability, to simplify the partitioning because the analogy between soft components and chips enables the hardware tools and the software tools to share a unified vision.
Table 2 comes from the work of G. Picard (Picard, 2004). It gives an insight of the different methods and the qualitative results of the comparison between them.

	Model of lifecycle	Requierements	Analysis	Design	Implementation	Test	Deploiement	Maintenance	Délivrables	Quality managment	Project managment
ADELFE *(Bernon et al., 2002)*	V	+	++	++	+	+	±	±	++	+	++
AAII *(Kinny et al., 1996)*	Waterfall	-	++	+	--	--	--	--	+	--	--
Aalaadin *(Ferber & Gutknecht, 1998)*	Waterfall	-	++	+	++	-	+	-	-	--	--
Cassiopée *(Drogoul & Collinot, 1998)*	Iterative	--	++	+	--	--	--	--	±	--	--
DESIRE *(Brazier et al., 2002)*	Waterfall	-	±	++	±	++	--	--	-	--	--
Gaia *(Wooldridge et al., 2000)*	Iterative	-	++	++	--	-	--	--	++	--	--
MaSE *(DeLoach et al., 2001)*	Waterfall	--	++	++	+	±	--	--	++	--	--
MASSIVE *(Lind, 2004)*	Incremental	+	++	++	+	+	++	±	+	-	-
MESSAGE *(Lind, 2001)*	Iterative	+	++	++	+	±	±	±	++	+	+
PASSI *(Chella et al., 2006)*	Incremental	+	++	++	+	±	++	±	++	--	--
Prométheus (Padgham et al., 2007)	Waterfall	-	++	++	+	-	--	--	+	--	--
Tropos *(Castor et al., 2004)*	Incremental	++	++	+	+	±	--	--	-	--	--
Voyelles *(Ricordel & Demazeau, 2000)*	Waterfall	-	++	++	+	+	+	--	--	--	--

DIAMOND	Spiral	+	++	++	++	(+)	++	+	+	(?)	(?)

++ : Properties are fully and explicitly supported
+ : Properties are taken care of in an indirect way
± : Properties are potentially Supported
--: Properties are not explicitly taken into charges
- : Properties are not supported

Table 2. Comparison synthesis of the multiagent methods

The criteria used in table 2 are:

- Requirements: Is the requirements gathering taken into account?
- Analysis: Is the analysis stage taken into account?
- Design: Is the design stage taken into account?
- Implementation: Is the implementation stage taken into account?
- Test: Is the testing process taken into account?
- Deployment: Is the deployment stage taken into account?
- Maintenance: Is the maintenance stage taken into account?
- Deliverables:
- Do the deliverables are clearly identified and associated with specific steps?
- Quality Management: Is the quality management taken into account?
- Project Management: Are the guidelines of conduct project are clear?

5.2 Models and notations

Multiagent method generally use notations and models from only one origin (Bernon et al., 2002) like UML (Mase , AAII, MESSAGE, PASSI). Other methods use many notation like TROPOS (notation i* coming from the knowledge engineering, A-UML (Koning et al., 2001) for interaction protocols and plan) or DESIRE (graph-based notation for knowledge modelling and specific hierarchical notation for tasks description). To cover all the phases of a lifecycle, we think like in (Herlea et al., 1999) that several formalisms are necessary for the different levels of abstraction.

DIAMOND begins by using UML use cases because they proved reliable for the definition of requirements. The interpretation of our use case diagrams is slightly different than their common use (as in (Bernon et al., 2002)) because actors are necessarily outdoor to the system or its entities. Moreover, an actor can not be in the interaction diagram (this would be amazing in a traditional use of UML use cases) in the case of physical interactions. These differences come from the usual software nature of applications.

In the analysis phase, we use context diagrams. These diagrams enable to see easily all the possible perception and the possible action of the agents. Another advantage is that they allow to see control flow between the physical part of an agent and its decisional part. In a word, context diagram allow to specify the external shell of the agents.

In the generic design phase, DIAMOND uses component as operational units as seen previously. In these components, we use finite state machines or a components set to describe the internal running. These formalisms enable to generate software code or hardware specifications in VHDL.

In this section, we compare our method with other multiagent methods (ADELPH, PASSI, MASE, GAIA, DESIRE, MASSIVE, MAMOSACCO etc.) In a first subsection we talk about lifecycle and stages. In the second subsection we focus on models and notations.

The methods multi-agents operating adopt mostly notations and models of a single origin (see table 3).

6. Conclusion

We work currently on the tool associated with the method that we propose. It is created using the Java language. The part which relates to the creation of agents with components, manual partitioning and automatic generation of code are operationnal.

	Requierement	**Analysis**	**Design**
ADELFE	UML diagrams (use case, sequence, collaboration)	UML diagrams (sequence, class), A-UML protocols	UML diagrams (class, paquetage, stéréotypes)
AAII		UML diagrams (collaboration, class)	UML object diagrams
Aalaadin		AGR organization diagram	A-UML diagrams
Cassiopée		FSM/dependency	
DESIRE		entity relationship diagram, FSM	Components
Gaia		Array, logic langage	
MaSE		UML sequence diagram	UML class diagrams
MAMOSACO	Arrays	UML class diagram, parametred Petri network, SADT actigram, OSSAD processing model	UML class diagrams, parametreed Petri networks
MASSIVE	UML use case diagrams	UML activity diagram	UML class diagrams
MESSAGE	UML use case diagrams	UML diagrams (class and activity), A-UML diagrams (collaboration)	UML class diagrams
PASSI	UML diagrams (use case, sequence), UML like packetage diagram	UML diagrams (sequence, class)	UML deployment diagrams
Prométheus		UML diagrams and A-UML diagrams	UML and A-UML diagrams
Tropos	i*	State diagram, A-UML protocols	
DIAMOND	UML diagrams (use case, sequence), textual specifications for the modes study, glossary	UML diagrams (sequence), A-UML protocols, context diagram (SART), entity relationship diagram (organisation)	FSM, components VHDL

Table 3. Notation used by these different methods

Our future work will be to improve the MASC tool (MultiAgent System Codesign) associated with the DIAMOND method. The agent design with components and the code generation in Java and C languages are operational. The VDHL specification generation is partially developed.

Very few works are addressing the problem of the analysis of self-organized embedded systems. This work proposes some innovative contributions in term of hybrid software/hardware multiagent lifecycle. It integrates in particular all the phases of the development from the analysis to the implementation. It introduces a multi-paradigm spiral lifecycle. It proposes components used as tools for integration, allowing software or hardware derivation. They enable a unified approach for all kinds of hybrid hardware/software multiagent systems.

7. References

Adams, J.; Thomas, D. (1996) The design of mixed hardware/software systems, *Proceedings of the 33st Conference on Design Automation*, pp 515-520, ISBN 0-89791-779-0, USA, June 1996, ACM Press.

Bernon, C.; Gleizes, M.-P.; Peyruqueou, S. & Picard, G. (2003), ADELPH: A methodology for adaptive multi-agent systems engineering., In: *Engineering Societies in the Agents World III*, page numbers 156-169, Springer Verlag, ISBN 3-540-14009-3, 2002,Spain.

Boehm, B. W. (1988). A spiral model of software development and enhancement. Computer, 21(5):61–72, 1988, IEEE Computer Society.

Brazier, F. M. T. ; Jonker, C. M. & Treur, J. (2002). Principles of component-based design of intelligent agents. *Data Knowledge Engineering,* Vol. 41, No. 1, April 2002, Elsevier, page numbers 1-27, ISSN 0169-023X.

Breuer, P. T.; Madrid, N.M.; Bowen, J. P.; France, R. B.; Larrondo-Petrie, M.M. & Kloos, C. D. (1999) Reasoning about vhdl and vhdl-ams using denotational semantics, *Proceedings of the Design, Automation and Test in Europe*, pp 346–352, ISBN 0-7695-0078-1, Germany, March 1999, IEEE Computer Society, Munich.

Carabelea, C.; Boissier, O. & Ramparany, F. (2003) Benefits and requirements of using multi-agent systems on smart devices, *Proocedings of 9th International Euro-Par Conference*, pp 1091-1098, ISBN 3-540-40788-X, Austria, August 2003, Springer Verlag, Klagenfurt.

Castor, A.; Pinto, R.; Silva, C. T. L. L. & Castro, J. (2004). Towards requirement traceability in tropos, *Proceedings of the Workshop em Engenharia de Requisitos*, pp 189–200, ISBN 950-658-147-9, Argentina, Dec. 2004, WER, Tandil.

Chella, A.; Cossentino, M., Sabatucci, L. & Seidita, V. (2006). Agile PASSI: An agile process for designing agents. *Computer Systems: Science & Engineering,* Vol. 21, No. 2, March 2006, In press, ISSN 0267-6192.

Antonio Chella, Massimo Cossentino, Luca Sabatucci, Valeria Seidita: Agile PASSI: An agile process for designing agents. Comput. Syst. Sci. Eng. 21(2): (2006)

Cossentino, M.; Sabatucci, L. & Chella, A. (2003). A possible approach to the development of robotic multi-agent systems, *Proceedings of the IEEE/ACM/WIC Conference on Intelligent Agent Technology*, pp 539–544, ISBN ISBN 0-7695-1931-8, Canada, 2003, Halifax.

Deguet, J.; Demazeau, Y. & Magnin, L. (2006). Elements about the emergence issue: A survey of emergence definitions, *Complexus*, Vol. 3, No. 1-3, 2006, pp. 24-31, ISSN 1424-8492.

DeLoach, S. A.; Wood, M. F. & Sparkman, C. H. (2001). Multiagent systems engineering. *International Journal of Software engineering and Knowledge Engineering,* Vol. 11, No. 3, June 2001, page numbers 231-258, ISSN 0218-1940

Dessalles, J.L.; Phan, D. (2005). Emergence in multi-agent systems: cognitive hierarchy, detection, and complexity reduction, *Proceedings of the 11th annual meeting of the Society of Computational Economics*, June 2005, Society of Computational Economics, University of Washington.

Demazeau, Y. (1995). From interactions to collective behavior in agent-based systems, *Proceedings of European Conference on Cognitive Science*, pp. 14-17, ISBN, France, Avril 1995, Saint-Malo

Drogoul, A & Collinot, A. (1998). Applying an agent oriented methodology to the design of artificial organizations: A case study in robotic soccer. *Journal on Agents and Multi-Agent Systems,* Vol. 1, No. 1, 1998, Kluwer Academic Press, page numbers 113-129, ISSN 1387-2532

Ferber, J. & Gutknecht, O. (1998). A Meta-Model for the analysis and design of organizations in multi-agent systems, *Proceedings of the 1998 International Conference on Multi-Agent Systems*, pp. 128-135, ISBN 0-8186-8500-X, France, July 1998, IEEE Computer Society, Paris.

Forrest, S. (1991). *Emergent computation,* The MIT Press, ISBN 978-0262560573, England.

Herlea, D. E.; Jonker, C. M.; Treur, J. & Wijngaards, N. J. E. (1999) Specification of Bahavioural Requirements within Compositional Multi-agent System Design, *Proceedings of 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pp 8–27, ISBN 3-540-66281-2, Spain, June 1999, Springer, Valencia.

Huang, H.-P.; Liang, C.-C & Lin C.-W. (2001) Construction and soccer dynamics analysis for an integrated multi-agent soccer robot system, *Natl. Sci. Counc. ROC(A)*, Vol. 25, No. 2, 2001, pp. 84-93.

Jamont, J.-P; Occello, M. (2007), Designing Embedded Collective Systems: The DIAMOND Multiagent Method, *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, pp. 91-94, ISBN 0-7605-3015-X, Greece, October 2007, IEEE Computer Society.

Jamont, J.-P; Occello, M. (2006), A Self-organized Energetic Constraints Based Approach for Modelling Communication in Wireless Systems, In: *Advances in Applied Artificial Intelligence*, page numbers 101-110, Springer Verlag, ISBN 3-540-35453-0, 2006, France.

Jamont, J.-P.; Occello, M. & Lagreze A. (2002). A multiagent system for the instrumentation of an underground hydrographic system, *Proceedings of IEEE International Symposium on Virtual and Intelligent Measurement Systems*, pp. 20-25, ISBN 0-7803-7344-8, USA, May 2002, IEEE Measurement and Instrumentation Society, Mt Alyeska Resort

Kinny, D.; Georgeff, M. & Rao, A. (1996). A methodology and modelling technique for systems of BDI agents, Agents Breaking Away: *Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-AgentWorld*, pp. 56-71, ISBN 3-540-60852-4, The Netherlands, January 1996, Springer-Verlag

Koning, J.-L.; Huget, M.-P.; Wie, J. & Wang, X. (2001). Extended Modeling Languages for Interaction Protocol Design, *Proceedings of the Second International Workshop on*

Agent-Oriented Software Engineering, pp. 68-83, ISBN 3-540-43282-5, Canada, May 2001, Springer, Montreal

Lind, J. (2004). *Interative Software Engineering for multiagent systems: The MASSIVE Method, Springer Verlag*, ISBN 3-540-42166-1, Berlin

Luo, J.; Xu, L.; Jamont, J.-P.; Zeng, L. & Shi Z. (2007). Flood decision support system on agent grid: method and implementation. *Enterprise Information Systems*, Vol. 1, No. 1, (November 2007) , Taylor and Francis, page numbers (1751-1757), ISSN 1751-1757.

Maña, A. & Rudolf, C.(2007). *Developing Ambient Intelligence,* Springer, ISBN 978-2-287-78543-6, Paris

Marcenac, P. (1996). Emergence of behaviors in natural phenomena agent-simulation. *Complexity International*, Vol. 3, 1996, ISSN 1320-0682.

Muller, J.-P. (2003), Emergence of collective behaviour and problem solving, In: *Engineering Societies in the Agents World IV*, page numbers 1-21, Springer, ISBN SBN 3-540-22231-6, 2003, England.

Occello, M. ; Demazeau, Y. & Baeijs C. (1998). Designing organized agents for cooperation in a real time context, *Proceedings of the first International Workshop of Collective Robotics*, pp. 25-73, ISBN 3-540-64768-6, France, March 1998, Springer-Verlag, Paris

Padgham, L.; Thangarajah, J. & WinikoffParunak, M., (2007). AUML protocols and code generation in the Prometheus design tool, *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp.270-271, ISBN 978-81-904262-7-5, Hawaii, May 2007, IFAAMAS.

Parunak, H. V. D. (2000). A practitioners? review of industrial agent applications. Autonomous Agents and Multi-Agent Systems, Vol. 3, No. 4, (2000) page numbers 389-407, ISSN 1387-2532

Picard, G. (2004). *Methodology for developping adaptive multi-agent systems and designing software with emergent functionality (PHD thesis),* Institut de Recherche en Informatique de Toulouse, France.

Ricordel, P.-M. & Demazeau, Y. (2000). From analysis to deployment: A multi-agent platform survey, *Proceedings of 1st International Workshop on Engineering Societies in the Agent World*, pp. 93-105, ISBN 3-540-41477-0, Germany, 2000, Springer-Verlag, Berlin

Russel, S. & Norvig P. (2002) *Artificial Intelligence : a Modern Approach – 2nd edition*. Prantice-Hall, ISBN 978-0137903955.

Steels, L. (1990). Cooperation between distributed agents through self-organisation, *Proceedings of IEEE Workshop on Intelligent Robots and Systems,* pp 8-14, ISBN 0-7803-8464-4, Japan, Jul. 1990, IEEE Robotics and Automation Society.

Wooldridge, M.; Jennings, N. R. & Kinny, D. (2000). The GAIA methodology for agent oriented analysis and design. *Journal on Agents and Multi-Agent Systems,* Kluwer Academic Publishers, Vol. 3, No. 3, September 2000, page numbers 285-312, ISSN 1387-2532

Wooldridge, M.-J. (1999). Intelligent agents. In: *Multiagent systems: A modern approach to Distributed Artificial Intelligence*, G. Weiss (Ed.), page numbers 27-79, MIT Press, ISBN 0-262-73131-2, 1999, England.

13

Content-based Image Retrieval Using Constrained Independent Component Analysis: Facial Image Retrieval Based on Compound Queries

Tae-Seong Kim[1] and Bilal Ahmed[2]
[1]*Department of Biomedical Engineering*
[2]*Department of Computer Engineering*
College of Electronics and Information
Kyung Hee University
Republic of Korea

1. Introduction

Visual information plays a crucial role in various domains, from medical diagnosis, journalism, crime-prevention to surveillance. Whereas domain specific images carry specific semantics, the problem of interpreting visual information becomes more complex when we talk of natural images. The maxim, 'A picture is worth a thousand words' explains this inherent problem very concisely. Indexing large databases of images for efficient retrieval is crucial for various domains such as journalism, biomedicine, forensics etc. Manual indexing of images in such large databases can be highly subjective and time consuming. In contrast, content-based image retrieval (CBIR) focuses on the development of efficient retrieval mechanisms based on image features or meta-data used for image annotation.

Conventional approaches to CBIR represent images in the form of image-based features. These features vary from global image descriptors such as color or intensity histogram to local ones such as shape and texture. These features along with their combinations have been used previously for CBIR. For example, in (Deng et al., 2001), a region-based color-descriptor, modelling the color values along with their percentages in the region, is proposed. Similarly in (Hadjidemetriou et al., 2004), multi-resolution histograms have been employed for the retrieval of textured images. In (Jeong et al., 2004), the extraction of color histograms through Gaussian mixture vector quantization has been proposed. In (Belongie et al., 2002) and (Petrakis et al., 2002) respectively, shape descriptors and shape matching algorithms have been proposed for image retrieval.

The use of low-level image features such as color histograms, shape, and texture attributes introduces a semantic gap (Chen et al., 2004). This semantic gap arises due to the inability of such low-level features to describe the objects and their inter-relations within the image. The use of such low-level features places the responsibility of achieving semantically coherent results on the user-interface. Various techniques of relevance feedback (Rui et al., 1998) have been introduced in this context. Whereas user feedback might be able to lower this gap, the overall procedure becomes subjective and requires a higher degree of user interaction.

Segmentation-based techniques for image retrieval have also been used for obtaining better shape, texture, and color descriptions of the image contents (Datta et al., 2005). The motivation behind this particular approach is that objects within an image can be segmented and used for querying the database to retrieve more semantically similar images. Various segmentation techniques, such as the Normalized Cuts (Shi & Malik, 2000), Mean Shift Procedures, and Expectation Maximization (Carson et al., 2002) algorithms have been used in image retrieval. Machine-learning approaches augmented with segmentation techniques have also been used. In (Wang et al., 2001), segmentation results augmented with fuzzy logic are used to obtain soft similarity measures. The problem of obtaining a semantically coherent segmentation of an image still remains an open research problem and higher dependency on segmentation-results is not desirable for achieving a semantically accurate retrieval performance.

From an image-retrieval point of view, facial images have attracted a lot of attention. Various machine learning and feature extraction techniques have been employed for the efficient retrieval of facial images. Earlier retrieval systems, such as the Photobook (Pentland et al., 1994), use Principal Component Analysis (PCA) for the retrieval of facial images. In (Liu, 2004), feature extraction through Independent Component Analysis (ICA) in a reduced PCA space is used for characterizing query images. The overall system comprises of classifying the input query image based on the nearest-neighbor rule using various similarity measures. A recent work on facial image retrieval by (Basak et al., 2006) has focused on representing facial images as a collection of local independent components. For this purpose, the query images are decomposed into a number of overlapping and non-overlapping windows to compute the independent components.

The use of multiple images as a compound query has not been explored in much detail. Conventional CBIR systems do not provide a mechanism through which a user can specify his search criterion through multiple examples. This is analogous to multi-word queries in search-engines: the specification of a compound query helps the system in retrieving the desired results with better accuracy. Similarly, when a user cannot find a single image which can specify his search criterion, he should be able to use multiple images to formulate his query. Multiple queries have been used in (Tahagogi et al., 1994): the approach taken is to find a combined result of the query by using the retrieved images corresponding to each query image independently. Similarly, (Basak et al., 2006) uses multiple facial images to retrieve images similar to the independent query-images as well as to their combinations.

In this work, we have devised a system which can cater for both single and multiple exemplar image retrieval. It does not decompose the query images or the database images to windows as in (Basak et al., 2006) or uses PCA for dimension reduction (Liu, 2004): thus the chances of any information loss are minimal. There is also no need to store additional feature information or the need for any offline learning as in (Basak et al., 2006). Our approach is centered on the idea of constrained ICA (cICA) (Lu & Rajapakse, 2005) which has the ability to extract specific independent components conforming to certain prior information (known as reference signals or images). Query images are provided to the constrained ICA algorithm as references, and the output of the constrained ICA algorithm specifies the contribution of each database image to the extracted component. Based on the magnitude of this contribution factor, the database images are ranked for retrieval.

The rest of the chapter gives details about the constrained ICA-based image retrieval system using multiple query images. Section 2 explains the constrained ICA framework and Section 3 describes the whole system in detail. Experimental results are given in Section 4. Finally, we conclude in Section 5.

2. Constrained ICA

Conventional ICA techniques perform blind-source separation (BSS) assuming a linear mixing model of the independent sources. If the observed values at a pixel location from a set of images are represented as $\mathbf{x} = (x_1,...,x_n)^T$ and the original sources as $\mathbf{s} = (s_1,...,s_m)^T$. ICA assumes that each $\mathbf{x}$ is a linear mixture of the original independent sources. Therefore,

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{1}$$

where $\mathbf{A}$ is the mixing matrix of $n \text{ x } m$. Conventional ICA algorithms aim at finding a demixing matrix $\mathbf{W}$ to recover all the ICs of the observed image such that $\mathbf{s} = \mathbf{W}\mathbf{x}$. In general, existing ICA algorithms find as many ICs as the number of observations (i.e., $n = m$). The user must manually identify which ICs represent which sources. The primary reason for this manual intervention is the inability of the ICA algorithms to calculate the energies or signs of the ICs. This may also lead to problems where the number of sources is less than the number of observations. Deflation-based ICA techniques (Cichoki et al., 1997); (Hyvärinen & Oja, 1996) have also been developed, but they also suffer from the arbitrary ordering of the extracted independent components.
Constrained ICA (Lu & Rajapakse, 2005) has been developed to find only those independent components which are of interest to the current task at hand. This is achieved by providing some prior knowledge about these ICs to the constrained ICA algorithm. This prior information may not be exact, but it could be the specification of statistical properties of the desired component or just a crude approximation (e.g., template). Therefore, if we have some *a priori* information about the desired sources, we can incorporate this information into constrained ICA. The constrained ICA algorithm uses this *a priori* information about the desired IC, encoded into a set of reference images, $\mathbf{r} = (r_1,...,r_l)^T$ to obtain a set of output IC images, $\mathbf{y} = (y_1,...,y_l)^T$ which contains statistically independent extracted sources. The closeness constraint can be written as,

$$g(\mathbf{w}) = \varepsilon(y_i, r_i) - \xi \leq 0 \tag{2}$$

where $\mathbf{w}$ is the weight vector to be learned, ε some closeness measure, and ξ an appropriate closeness threshold parameter. The measure of closeness can take any form, such as the mean squared-error (MSE), correlation, or any other suitable measure. The number of reference signals determines the number of independent components to be extracted from the complete set of observations. The final mathematical model for constrained ICA can be represented as,

$$\text{maximize} \sum_{i=1}^{l} J(y_i) \tag{3}$$
$$\text{subject to } g(\mathbf{W}) \leq 0, h(\mathbf{W}) = 0$$

where,

$$J(y_i) \approx \rho[E\{G(y_i)\} - E\{G(v)\}]^2 \tag{4}$$

is the one-point contrast function for ICA introduced in (Hyvärinen et al., 2001). ρ is a positive constant, $G(\cdot)$ a non-quadratic function, and v a zero mean and unit variance Gaussian random variable. $h(\mathbf{W})$ constrains the output component to have unit variance. Equation (4) is a constrained optimization problem and can be solved by the augmented Lagrangian functions.

3. Constrained ICA-based facial image retrieval

Viewing it from another perspective, the constrained ICA framework can be used for specifying the type of information we would like to extract from huge amounts of data. The reference image(s) can be formulated as the query image(s) specified by the user and as the accuracy of the extracted information depends upon the accuracy of the provided references. In our case, the image(s) provided by the user would serve this purpose, and point the constrained ICA algorithm in the appropriate direction. The overall system architecture is depicted in Fig. 1.

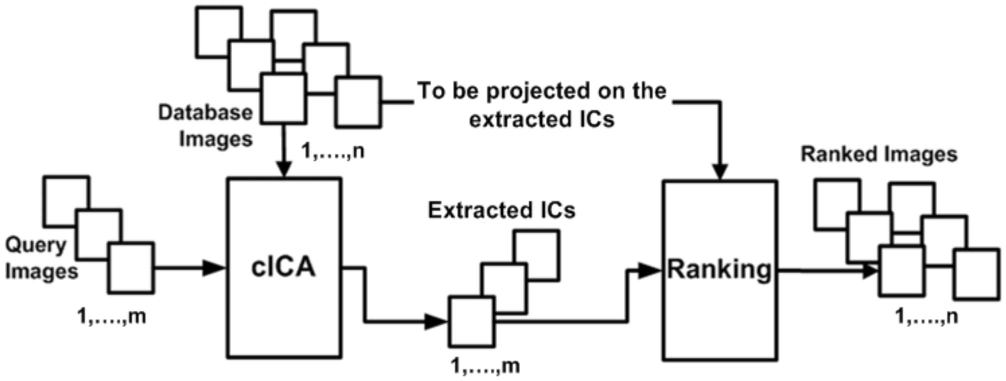


Fig. 1. Overview of the constrained ICA-based CBIR system.

Since constrained ICA extracts components y_i from the given set of observations corresponding to the provided reference image(s), we can ascertain the contribution of each observation by reconstructing it from the extracted component. The reconstruction procedure involves the estimation of the mixing matrix $\mathbf{A}$ and the reconstruction of the entire set of observations. Consider that we have n observations $\mathbf{x} = (x_1,...,x_n)^T$ and m extracted sources $\mathbf{s} = (s_1,...,s_m)^T$ where $m << n$. The mixing matrix pertaining to the extracted sources with respect to the entire set of observations can be estimated using,

$$\mathbf{A} = \mathbf{x}\mathbf{s}^+ \tag{5}$$

where $\mathbf{s}^+$ is the pseudoinverse of the extracted sources. Furthermore the reconstruction of $\mathbf{x}$ can be done using,

$$\mathbf{x}_R = \mathbf{As} \tag{6}$$

where $\mathbf{x}_R$ is the set of reconstructed observations.

Once all images have been reconstructed with the extracted IC images, we need to estimate how well each image has been reconstructed from the extracted sources. This involves the comparison of each reconstructed image with its original image. Simple measures of similarity such as correlation or mutual-information can be used. In our case we have used correlation to determine the similarity between the original and the reconstructed images.

4. Experimental results

We have conducted extensive experiments on a publicly available facial-image database, the ORL face database (Samaria & Harter, 1994). The ORL database contains ten facial images of 40 individuals with varying pose, expressions, and spectacles. The images were scaled to 64×64 with no pre-processing or feature extraction.

For the purpose of evaluating our retrieval system, we have divided the queries into two categories: homogeneous and heterogeneous queries. Homogeneous queries contain the images of the same person with different pose, expression, or occlusion. Heterogeneous queries are composed of images of different subjects. The main motivation behind this formulation is to bring out the essence of fusing information from different independent images and evaluating the semantic coherence of the retrieved images.

4.1 Homogeneous queries

In order to retrieve the facial images of a single person from the database under varying pose and occlusion conditions (i.e., wearing spectacles), a single example might not be enough. The same is also true if the database has different expressions and scale. Fig. 2 (*left*) shows the results of using a single query. The database contains ten images of each individual: with a single query image, our system has been able to retrieve eight images in the top ten retrieved images. The images, which have been left out of the top ten: the image at (3rd row, 1st column) and image at (3,2) in Fig. 2 (*left*), have the same individual but with his head tilted to the right side. The query image given in Fig. 2 (*left*) was unable to describe the features present in these left-out images.

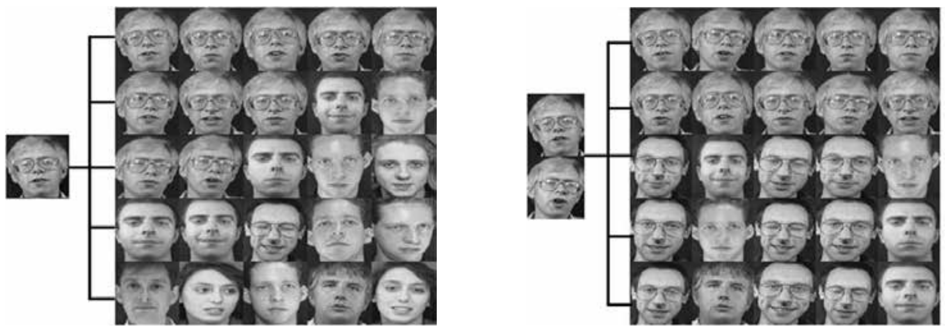


Fig. 2. Constrained ICA-based CBIR applied to the ORL database. (*Left*) A result with a single query: here the system acts as a face recognition system. (*Right)* A result using two images of the same individual with different pose.

In Fig. 2 (*right*), we have used two query images: one depicts the individual with a left tilt whereas the other depicts the same pose but in the opposite direction. All the ten relevant images have been retrieved from the database and have the highest ranking, as can be seen from the results. This particular case shows the fact that the constrained ICA-based retrieval technique is able to fuse features from two independent images and retrieve images in which the subject has a straight pose.

4.2 Heterogeneous queries

Fig. 3 shows the results obtained for two heterogeneous queries. In the first query depicted in Fig. 3 (*left*), two images of two different individuals have been used. One of them is wearing spectacles whereas the other has none. In the retrieved images, we see that the initial nine images correspond to the two individuals, where images of the second subject wearing spectacles are also given a higher rank. Similarly, after the two top rows, the system has retrieved images of individuals with and without spectacles and bearing some facial similarity to the individuals depicted in the query images.

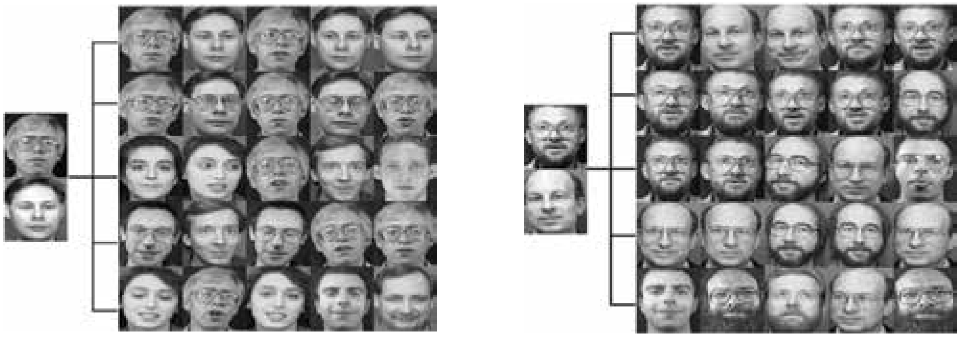


Fig. 3. Results for heterogeneous queries.

In the second case shown in Fig. 3 (*right*), again images of two different individuals are used. This time, the top query image has an individual who has a beard and spectacles. Whereas the other individual is clean shaven and has no spectacles. The images retrieved by the system not only contain the individuals present in the query but also their various combinations: persons having both beard and spectacles (the same as the individual in the top query images), persons having only beard, persons wearing only spectacles with no beard, and persons having none of these (corresponding to the individual depicted in the lower query image).

4.3 Image retrieval of covered faces

In the case of image retrieval, query images have a profound effect on the output of the system. It could be the case that the images available at the time of query formulation contain only partial information about the target image(s). As an example, consider the case of querying the database when the available face images are covered with some objects such as sunglasses or muffler. Fig. 4 *(left)* depicts a case of such a query in which the lower-half of the subject's face is covered with muffler. As the results show, the retrieved images are those in which all the subjects have their face covered in the same manner.

This situation can be alleviated by fusing information from another face image. The constrained-ICA based image retrieval framework allows for such information fusion through the use of multiple-query images. Fig. 4 *(right)* shows the results for such a query: where along with the covered face, we now provide the system with another face image without occlusion. Based on this partial information, the system is able to retrieve the desired face image at (3rd row, 4th column). The system also retrieves the same subject, wearing sunglasses at (4,3).

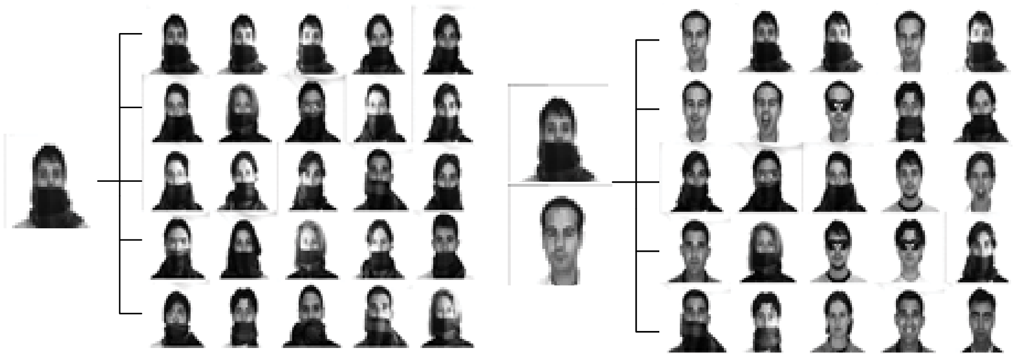


Fig. 4. Constrained ICA-based CBIR applied to the Aleix Face database (Martinez & Benavente, 1998). Two results are shown with a partially covered query image (*left*) and augmented query images with a normal face (*right*).

4.4 Performance analysis

We conducted one hundred simulations of the system with random query formulations for the homogeneous query case. A hard similarity evaluation was used: only the retrieved images pertaining to the same individual as depicted in the query were considered relevant as opposed to (Basak et al., 2006) where it was assumed that user feedback is available. Simple measures of *precision* (Baeza-Yates & Ribeiro-Neto, 1999) and *recall* (Baeza-Yates & Ribeiro-Neto, 1999) have been used to evaluate the efficiency of the system:

$$precision = {}^{N_{RL}}\!/_{N_R} \tag{7}$$

$$recall = {}^{N_{RL}}\!/_{N_{RD}} \tag{8}$$

where N_{RL} is the number of relevant images in the retrieved images, N_R the total number of retrieved images, and N_{RD} the total number of relevant images in the database. In the case of Fig. 2 (a), Precision = 10/25 and Recall = 10/10. Note that, when N_R equals N_{RD} the two measures become equal. This is the break-even point of the system and indicates its overall accuracy.

The evaluation measures for queries consisting of one, two, and three images are shown in Fig. 5. In the figure, T1, T2, and T3 represent the break-even points of the system for the queries formulated from one, two, and three images respectively. In the case of single-image queries, the system has achieved an accuracy of 76%. Whereas, in the case of compound queries composed of two and three images, this accuracy increases to 80% and 90% respectively. In contrast to the conventional systems, the constrained ICA-based retrieval system achieves this higher level of performance without any feature-extraction and offline-learning.

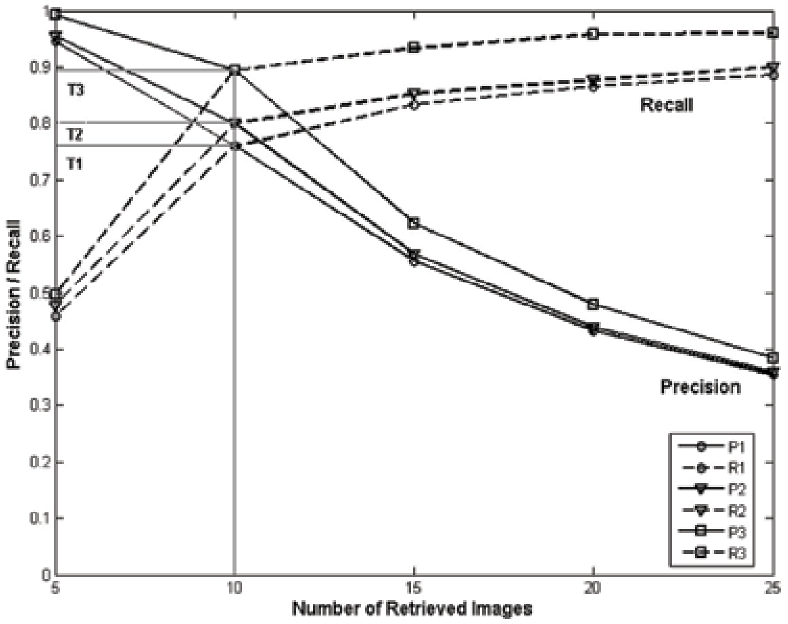


Fig. 5. Performance evaluation on homogeneous queries. The graph shows the precision (P) and recall (R) values for queries formulated with 1, 2, and 3 images.

5. Conclusion

In this work, we have proposed a new technique of facial image retrieval based on constrained ICA. Our technique requires no offline learning, pre-processing, and feature extraction. The system has been designed so that none of the user-provided information is lost, and in turn more semantically accurate images can be retrieved. As our future work, we would like to test the system in other domains such as the retrieval of chest x-rays and CT images.

6. Acknowledgement

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Advancement) (IITA-2008-(C1090-0801-0002)).

7. References

Baeza-Yates, R. A. & Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ISBN: 020139829X, Addison-Wesley.

Basak, J.; Bhattacharya, K. & Chaudhury, S. (2006). Multiple Exemplar-Based Facial Image Retrieval Using Independent Component Analysis. *IEEE Transactions on Image Processing*, 15, 12, (December 2006) 3773–3783, ISSN: 1057-7149.

Belongie, S.; Malik, J. & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24, 4, (April 2002) 509–522, ISSN: 0162-8828.

Carson, C.; Belongie, S., Greenspan, H. & Malik, J. (2002). Blobworld: Image segmentation using expectation-maximization and its applications to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24, 8, (August 2002) 1026–1038, ISSN: 0162-8828.

Chen, Y.; Li, J. & Wang, J. Z. (2004). *Machine Learning and Statistical Modeling Approaches to Image Retrieval*. ISBN: 1402080344, Kluwer Academic Publishers.

Cichoki, A.; Thawonmas, R. & Amari, S. (1997). Sequential blind signal extraction in order specified by stochastic properties. *Electronics Letters*, 33, 1, (Janunary 1997) 64–65, ISSN: 0013-5194.

Datta, R.; Li, J. & Wang, J. Z. (2005). Content-Based image retrieval - approaches and trends of the new age. *Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval*. Singapore, November 2005.

Deng, Y.; Manjunath, B. S., Kenney, C., Moore, M. S. & Shin, H. (2001). An efficient color representation for image retrieval. *IEEE Transactions on Image Processing*, 10, 1, (January 2001) 140–147, ISSN: 1057-7149.

Hadjidemetriou, E.; Grossberg, M. D. & Nayar, S. K. (2004). Multiresolution histograms and their use for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 7, (July 2004) 831–847, ISSN: 0162-8828.

Hyvärinen, A.; Karhunen, J. & Oja, E. (2001). *Independent Component Analysis*. ISBN: 047140540X, Wiley-Interscience.

Hyvärinen, A. & Oja, E. (1996). Simple neuron models for independent component analysis. *International Journal of Neural Systems*, 7, 6, (February 1996) 671–687, ISSN: 0129-0657.

Jeong, S.; Won, C.S. & Gray, R. M. (2004). Image retrieval using color histograms generated by Gaussian mixture vector quantization. *Computer Vision and Image Understanding*, 9,1-3, (April 2004) 44–66, ISSN: 1077-3142.

Liu, C. (2004). Enhanced independent component analysis and its application to content based face image retrieval. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 34, 2, (April 2004) 1117–1127, ISSN: 1083-4419.

Lu, W. & Rajapakse, J. C. (2005). Approach and applications of constrained ICA. *IEEE Transactions on Neural Networks*, 16, 1, (January 2005) 203–212, ISSN: 1045-9227.

Martinez, A. M. & Benavente, R. (1998). The AR Face Database. *CVC Technical Report #24*, June 1998.

Pentland, A.; Picard, R. W. & Sclaroff, S. (1994). Photobook: tools for content-based manipulation of image databases. *In Proc. SPIE Storage and Retrieval for Image and Video Databases II*, San Jose, CA, USA, February 1994.

Petrakis, E. G. M.; Diplaros, A. & Milios, E. (2002). Matching and retrieval of distorted and occluded shapes using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24, 11, (November 2002) 1501 – 1516, ISSN: 0162-8828.

Rui, Y.; Huang, T. S., Ortega, M. & Mehrotra, S. (1998). Relevance feedback: A power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8, 5, (September 1998) 644–655, ISSN: 1051-8215.

Samaria, F. & Harter, A. (1994). Parameterization of a stochastic model for human face identification. *Proceedings of 2nd IEEE Workshop on Applications of Computer Vision*, Sarasota FL, December 1994.

Shi, J. & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, 8, (August 2000) 888–905, ISSN: 0162-8828.

Tahaghoghi, S. M. M.; Thom, J. A. & Williams, H. E. (2001). Are two pictures better than one?. *Proc. of the 12th Australasian Database Conference*, pp. 138–144, Gold Coast, Queensland, Australia, January 2001.

Wang, J. Z.; Li, J. & Wiederhold, G. (2001). SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23, 9, (September 2001) 947–963, ISSN: 0162-8828.

14

Text Classification Aided by Clustering: a Literature Review

Antonia Kyriakopoulou
Athens University of Economics and Business
Greece

1. Introduction

Supervised and unsupervised learning have been the focus of critical research in the areas of machine learning and artificial intelligence. In the literature, these two streams flow independently of each other, despite their close conceptual and practical connections. In this work we exclusively deal with the text classification aided by clustering scenario. This chapter provides a review and interpretation of the role of clustering in different fields of text classification with an eye towards identifying the important areas of research. Drawing upon the literature review and analysis, we discuss several important research issues surrounding text classification tasks and the role of clustering in support of these tasks. We define the problem, postulate a number of baseline methods, examine the techniques used, and classify them into meaningful categories.
A standard research issue for text classification is the creation of compact representations of the feature space and the discovery of the complex relationships that exist between features, documents and classes. There are several approaches that try to quantify the notion of information for the basic components of a text classification problem. Given the variables of interest, sources of information about these variables can be compressed while preserving their information. Clustering is one of the approaches used in this context. In this vein, an important area of research where clustering is used to aid text classification is the area of dimensionality reduction. Clustering is used as a ***feature compression and/or extraction method***: features are clustered into groups based on selected clustering criteria. Feature clustering methods create new, reduced-size event spaces by joining similar features into groups. They define a similarity measure between features, and collapse similar features into single events that no longer distinguish among their constituent features. Typically, the parameters of the cluster become the weighted average of the parameters of its constituent features. Two types of clustering have been studied: i) one-way clustering, i.e. feature clustering based on the distributions of features in the documents or classes, and ii) co-clustering, i.e. clustering both features and documents.
A second research area of text classification where clustering has a lot to offer, is the area of ***semi-supervised learning***. Training data contain both labelled and unlabelled examples. Obtaining a fully labelled training set is a difficult task; labelling is usually done using human expertise, which is expensive, time consuming, and error prone. Obtaining unlabelled data is much easier since it involves collecting data that are known to belong to

one of the classes without having to label them. Clustering is used as a method to extract information from the unlabelled data in order to boost the classification task. In particularly clustering is used: i) to create a training set from the unlabelled data, ii) to augment the training set with new documents from the unlabelled data, iii) to augment the dataset with new features, and iv) to co-train a classifier.

Finally, ***clustering in large-scale classification problems*** is another major research area in text classification. A considerable amount of work is done on using clustering to reduce the training time of a classifier when dealing with large data sets. In particular, while SVM classifiers (see (Burges, 1998) for a tutorial) have proved to be a great success in many areas, their training time is at least $O(N^2)$ for training data of size N, which makes them non favourable for large data sets. The same problem applies to other classifiers as well. In this vein, clustering is used as a down-sampling pre-process to classification, in order to reduce the size of the training set resulting in a reduced dimensionality and a smaller, less complex classification problem, easier and quicker to solve. However, it should be noted that dimensionality reduction is not accomplished directly using clustering as a feature reduction technique as discussed earlier, but rather in an indirect way through the removal of training examples that are most probably not useful to the classification task and the selection of the most representative redundant training set. In most of the cases this involves the collaboration of both clustering and classification techniques.

The chapter is organized as follows: the next section presents a review of the literature on text classification aided by clustering. It provides a comprehensive summary of the alternative views and applications of clustering discussed above and their implications for text classification. A broader perspective on clustering and text classification research is then provided by discussing important research themes that emerge from the review of the literature and by classifying them into meaningful concept groups. We conclude by pointing out open issues and limitations of the techniques presented.

2. The literature review

2.1 Clustering as a feature compression and/or extraction method

Clustering as a feature compression and/or extraction method includes: i) one-way clustering, and ii) co-clustering.

2.1.1 One-way clustering (clustering features)

In text classification using one-way clustering, a clustering algorithm is applied prior to a classifier to reduce feature dimensionality by grouping together "similar" features into a much smaller number of feature clusters, i.e. clusters are used as features for the classification task replacing the original feature space. A crucial stage in this procedure is how to determine the similarity of features. Three main clustering methods have been applied in the literature: information bottleneck, distributional clustering, and divisive clustering.

An important feature clustering method that formulates a principle for the extraction and efficient representation of relevant information is the ***information bottleneck (IB)*** method (Tishby et al., 1999). The objective of the IB method is to extract the information from one variable X that is relevant for the prediction of another variable Y. In other words, the method finds an efficient compressed representation of the variable X, denoted X', such that the predictions of *Y* from *X* through *X′* is as close as possible to the direct prediction of *Y* from *X*. The compactness of the representation is determined by the mutual information *I(X;X′)* while the quality of the clusters is measured by the fraction of information they capture about *Y*, that is *I(X′;Y)/I(X;Y)*. Obviously there is a trade-off between compressing the representation and preserving meaningful information. The desirable is to keep a fixed amount of meaningful information about the relevant variable, *Y*, while minimizing the number of bits from the original variable *X* (maximizing the compression). In an alternative ***agglomerative implementation of the IB*** method, (Slonim & Tishby, 1999) attain maximum mutual information per cluster between feature data and given categories. This implementation can be considered as a bottom-up hard implementation of the original top-down soft hierarchical IB method. They demonstrate the algorithm on a subset of 20Newsgroups corpus, achieving compression by 3 orders of magnitude while maintaining about 90% of the original mutual information. The IB clustering method with its variations is used in the context of text classification by many authors. In this vein the classifier is applied in a reduced space where features represent clusters.

More specifically, in (Slonim & Tishby, 2001) the IB clustering method is used together with the Naive Bayes (NB) classifier. First, the feature clusters that preserve the information about the classes as much as possible are found using the agglomerative IB method. Then these clusters are used to represent the documents in a new, low dimensional feature space and the NB classifier is applied on this reduced space. Results from 20Newsgroups corpus show that when the size of the data sets is large, using feature clusters does not improve significantly the classification performance. However, when a small sample training set is used the method yields a significant improvement in classification accuracy, from 5% to 18%, compared to using the original feature space. (Verbeerk, 2000a, 2000b) applies the minimum description length (MDL) (Rissanen, 1989) principle to the agglomerative algorithm of (Slonim and Tishby, 2001) in order to define the number of clusters to be used for the classification task.

(Bekkerman et al., 2001, 2003) compare two classification schemes based on two representations: the simple, typical bag-of-words (BOW) representation (Salton & McGill, 1983) together with mutual information feature selection, and a representation that is based on feature clusters computed via the IB method. The comparison is performed over 20NG, Reuters-21578 and WebKB with SVMs used for the classification task. The results of the experiments are contradictory revealing a sensitivity of the algorithm to the datasets.

(Mubaid & Umair, 2006) use the IB clustering method with a least squares (Felici & Truemper, 2002) classifier. The method has been tested with the WebKB, 20NG and Reuters-21578 datasets and is compared against SVM. The experimental results show that the performance of the method is equally good and in some cases outperforms SVM, especially when there is limited training data.

(Baker & McCallum, 1998) apply ***distributional clustering*** as a feature clustering method for text classification. Distributional clustering (Pereira et al, 1993) is a special case of the general IB clustering algorithm as it is shown in (Slonim and Tishby, 2001). The similarity between two features, f_t and f_s , is measured as the similarity between the class variable *C* distribution they induce: $P(C|f_t)$ and $P(C|f_s)$. In the case of text classification, the similarity of two features is the similarity between their joint distributions with the category variable. For clustering this means that features with similar distributions over the classes (should) belong to the same cluster. Intuitively, if two different features have similar distributions over the classes, they will play a similar role in the classification process, and thus might as well be clustered together. Using a Naïve Bayes classifier for the classification task, they compare their method with feature selection methods such as Latent Semantic Indexing, class-based clustering, mutual information, and Markov-blanket-based feature selection (Koller & Sahami, 1996). Their results show that distributional clustering outperforms the other methods by drastically reducing the number of features, achieving compression by 3 orders of magnitude, while loosing only 2% classification accuracy. An interesting outcome concerns the application of a feature selection method prior to the feature clustering method. It actually improves the feature clustering method, suggesting that there is place for combinations of the two methods.

(Dhillon et al, 2003a) propose an information-theoretic feature clustering algorithm, termed as ***divisive clustering***, and apply it to text classification. The method derives a global objective function that explicitly captures the optimality of feature clusters in terms of a generalized Jensen-Shannon divergence (Lin, 1991) between multiple probability distributions. Then a fast, divisive algorithm that monotonically decreases this objective function value is applied. The algorithm has many good qualities. Is optimises over all clusters simultaneously and it is much faster than the agglomerative strategies proposed by (Baker & McCallum, 1998) and (Slonim & Tishby, 2001) obtaining better feature clusters. Experiments using the Naive Bayes and SVM classifiers on the 20 Newsgroups and Dmoz data sets show that divisive clustering improves classification accuracy especially at lower number of features. When the training data is sparse, divisive clustering achieves higher classification accuracy than the maximum accuracy achieved by feature selection strategies such as information gain and mutual information.

(Lavelli et al., 2004) carry out experiments on feature classification tasks (i.e. grouping together features according to their meaning into prespecified classes) and feature clustering tasks in order to compare the two representations. Also, (Lewis, 1992) studies the properties of clustered feature representations on a text classification task. See (Jain et al., 1988) for a comprehensive survey on one-way clustering.

2.1.2 Co-clustering (clustering features and documents)

Using co-clustering in text classification, a two-stage procedure is usually followed: feature clustering and then document clustering. In this way a reduction for both dimensions is attained.

The ***double-clustering (DC)*** algorithm (Slonim & Tishby, 2000) is a co-clustering two-stage procedure based on the IB method. Intuitively, in the first stage of DC, feature clustering generates coarser pseudo features, which reduce noise and sparseness that might be exhibited in the original feature space. Then, in the second stage, documents are clustered as distributions over the "distilled" pseudo features, and therefore generate more accurate document clusters. An extension of the DC algorithm, the so called ***Iterative Double Clustering (IDC)*** (Yaniv & Souroujon, 2001) applies the DC algorithm in an iterative manner. Whenever the first DC iteration succeeds in extracting a meaningful structure of the data, a number of the next consecutive iterations can continually improve the clustering quality. This is achieved due to the generation of progressively less noisy data representations. Experiments conducted on text classification tasks indicate that IDC outperforms DC and competes even SVM when the training set is small. The works of (Slonim & Tishby, 2000), (Slonim et al., 2001), (Yaniv & Souroujon, 2001) use heuristic procedures to cluster documents and features independently using an agglomerative algorithm.

(Dhillon et al, 2002, 2003b) on the other hand, propose an ***information-theoretic co-clustering*** algorithm that intertwines both row (feature) and column (document) clustering. The algorithm starts with a random partition of rows, *X*, and columns, *Y*, and computes an approximation *q(X,Y)* to the original distribution *P(X,Y)* and a corresponding compressed distribution by co-clustering rows and columns intertwined, i.e. the row-clustering incorporates column-clustering information and vice versa. The algorithm iterates until it almost accurately reconstructs the original distribution, discovers the natural row and column partitions and recovers the ideal compressed distribution. Experiments conducted demonstrate the efficiency of the algorithm especially in the presence of sparsity.

(Dai et al., 2007) extend the co-clustering algorithm of (Dhillon et al., 2002, 2003b) and present a ***co-clustering classification algorithm*** (CoCC) that focuses on classifying documents across different text domains. There is a labelled data set D_i from one domain, called *in-domain*, and an unlabelled data set D_o from a related but different domain, called *out-of-domain*, that is to be classified. The two datasets are drawn from different distributions, since they are from different domains. The algorithm is based on two assumptions. First, the set *C* of class labels in D_i prescribes the labels to be predicted in D_o. Second, even though the two domains have different distributions, they are similar in the sense that similar words describe similar categories, thus, the probability of a class label given a word is very close in the two domains. The algorithm applies co-clustering between all features and out-of-domain documents (new tasks) in D_o. Feature clustering is constrained by the labels of in-domain (old) documents *Di*. The feature clustering part in both domains serves as a bridge. For the classification task, each out-of-domain cluster is mapped to a corresponding class label based on the correlation with the document categories in D_i.

The idea of clustering features and documents to improve text classification is also pursued in (Takamura & Matsumoto, 2002; Takamura, 2003). They empirically show that the assumption that documents in the same category are generated from an independent identical distribution is inaccurate, and propose a new method called ***two-dimensional clustering*** to alleviate this problem. According to this method, training examples are first clustered so that the i.i.d. assumption is more likely to be true and features are also clustered in order to deal with the data-sparseness problem caused by the high dimensionality of the feature space. Two classifiers (NB and SVM) are trained on the training examples of each cluster and the testing examples are classified and assigned the label of the class of the cluster (all training examples in each cluster are supposed to have the same class label). The comparison of the method with distributional clustering (Baker & McCallum, 1998) and feature clustering on Reuters-21578 and 20NG shows promising results.

Table 1 summarizes the methods presented in this section.

2.2 Clustering in semi-supervised classification

Clustering in semi-supervised classification is used as a method to extract information from the unlabelled data in order to boost the classification task. In particularly clustering is used: i) to create a training set from the unlabelled data, ii) to augment the training set with new documents from the unlabelled data, iii) to augment the dataset with new features, and iv) to co-train a classifier.

Goal	Authors	Clustering method
One-way clustering: cluster feature space and replace it with a feature cluster representation	(Baker & McCallum, 1998)	Distributional clustering
	(Slonim & Tishby, 2001)	IB
	(Verbeerk, 2000a, 2000b)	Agglomerative IB
	(Bekkerman et al., 2001, 2003)	Agglomerative IB
	(Mubaid & Umair, 2006)	IB
	(Dhillon et al, 2003a)	Divisive clustering
Co-clustering: cluster both features and documents	(Yaniv & Souroujon, 2001)	Iterative double clustering
	(Dhillon et al, 2002, 2003b)	Information-theoretic co-clustering
	(Dai et al., 2007)	Co-clustering classification
	(Takamura & Matsumoto, 2002);(Takamura, 2003)	Two-dimensional clustering

Table 1. Clustering as a feature compression and/or extraction method

2.2.1 Create a training set from the unlabelled data

(Fung and Mangasarian, 2001) propose a model for classifying two-class unlabelled data, called ***clustered concave semi-supervised SVM (CVS^{3}VM)***. First, a *k*-median clustering algorithm finds *k* cluster centres for the unlabelled examples such that the sum of distances between each example and the closest cluster centre is minimized. Then, examples within a certain distance from these *k* cluster centres are treated as representative examples of the clusters, and hence of the overall dataset, and are given to an expert or oracle to label. Finally, a linear SVM is trained using this small sample of labelled data. The model is effectively compared to other methods.

(Li et al., 2004) follow a similar approach where a *k*-means clustering algorithm is used to cluster the unlabelled data into a certain number of subsets and to assign corresponding cluster labels. Then, an SVM classifier is trained on this labelled set.

2.2.2 Augment the training set with new documents from the unlabelled data

The ***clustering based text classification (CBC)*** approach (Zeng et al., 2003) improves classification performance by using unlabelled data, *U*, to augment the training, labelled data, *L*. According to this method a clustering algorithm is first applied to *L*. For each class, the centroids of the labelled data are computed and used as the initial centroids for *k*-means. The *k* value for *k*-means is set to the number of classes in the classification task. Accordingly, the label of each centroid is equal to the label of the corresponding examples of each class. Then, *k*-means runs for both *L* and *U* and *k* clusters are created. The most confident examples from each cluster (i.e. the ones nearest to the cluster's centroid) are added to *L*. This is considered to be a soft-constrained version of *k*-means because the constraints are not based on exact examples but on their centroid, thus reducing the bias in *L*. Finally, the augmented *L* and the rest of *U* are used to train and test a Transductive SVM (TSVM) classifier. Their experimental results demonstrate that CBC outperforms existing algorithms, such as TSVMs and co-training, especially when the size of the labelled dataset is very small.

(Chapelle et al., 2002) propose a framework to incorporate unlabelled data in a kernel classifier based on the "cluster assumption", i.e. nearby points are likely to have the same class label, and two points are likely to have the same class label if they belong to the same cluster. Using spectral methods (Spielman & Teng, 1996; Ng et al., 2002) they show how to design kernels such that the induced distance is small for points in the same cluster and large for points in different clusters. This representation with the points naturally clustered, is then used to train a discriminative learning algorithm. The testing set, if available during training, can be considered as unlabelled data; therefore spectral clustering is applied to training, unlabelled and testing data. Otherwise, an approximation of each testing example as a linear combination of the training and unlabelled data is computed. The experiments show encouraging results. The algorithm is applicable to a purely supervised learning task.
(Zhou et al., 2003) also base their method on the "cluster assumption" and apply spectral clustering to represent the labelled and unlabelled data. The keynote of the method is to let every labelled point in the representation iteratively spread its label information to its neighbours until a global stable state is achieved. Then, the label of each unlabelled point is set to be the class of which it has received most information during the iteration process. The algorithm demonstrates effective use of unlabelled data in experiments including digital recognition and text categorization.

2.2.3 Augment the dataset with new features

Unlike direct methods like CBC, which label the unlabelled data, the technique of (Raskutti et al., 2000a), augments the feature space with new features derived from clustering the labelled and unlabelled data. A non-hierarchical single-pass clustering algorithm is used to cluster labelled and unlabelled examples. In order to derive only the useful information from the clusters, the clusters are sorted by their sizes, and the largest *N* clusters are chosen as representatives of the major concepts. Each cluster contributes the following features to the feature space of the labelled and the testing examples: i) a binary feature indicating if this is the closest of the *N* clusters, ii) similarity of the example to the cluster's centroid, iii) similarity of the example to the cluster's unlabelled centroid, i.e. the average of the unlabelled examples that belong to the cluster, and iv) for each class in the labelled set, similarity of the example to the cluster's class *l*-centroid defined as the average of the examples in class *l* that belong to this cluster. The clusters are thought of as higher level "concepts" in the feature space, and the features derived from the clusters indicate the similarity of each document to these concepts. The unlabelled data are used to improve the representation of these concepts. They evaluate the method using SVM classifiers on well-known corpora, and find significant improvements in the classification performance.
In (Kyriakopoulou & Kalamboukis, 2007) the training and testing sets are augmented with new features derived from clustering without using unlabelled data. Consider a *k*-class categorization problem, (k>=2), with a labelled *l*-training sample $\{(x_1, y_1),\ldots,(x_l, y_l)\}$ of feature vectors $x \in R^n$ and corresponding labels $y_i \in \{1, \ldots, k\}$, and an unlabelled *m*-testing sample $\{(x_1^*,\ldots,x_m^*\}$ of feature vectors. The approach consists of three steps: clustering, expansion and classification step. In the clustering step, the number of clusters is chosen to be equal to *k*, i.e. the predefined number of classes. A divisive clustering algorithm with repeated bisections is selected to cluster both training and testing sets. In the expansion step, each cluster contributes one *meta*-feature to the feature space of the training and testing sets: given the total *n* features that are used in the representation of the *l*+*m* feature vectors, and

the k clusters derived from the clustering step, create *meta*-features $x_{n+1},\ldots,x_{n+k}$. A document x in the cluster C_j is characterized by the *meta*-feature x_{n+j}. Finally, in the classification step, linear SVM/transductive SVM classifiers are trained on the expanded training set and classify the expanded testing set. Evaluation of this approach using several widely used corpora indicates that it is extremely useful improving the classifier's performance especially when the number of the training examples is very small. The algorithm has also been successfully used in a spam-filtering setting (Kyriakopoulou & Kalamboukis, 2006). Also, it can be directly applied to a purely semi-supervised task using unlabelled data as an additional source of information.

In (Takamura, 2003) given the co-occurrences of features and documents of the training set, the features are first hard clustered. Let H be the reduced matrix resulting from clustering. The relation between a feature vector d and its reduced vector s is $Hd=s$. Next, the two vectors are concatenated into a vector d'. Then, the testing set is classified with SVM using d' as input. Takamura explains how the expansion of the feature space is equivalent to using a special kernel in the original feature space, where the form of the mapping to a higher dimensional space depends on the given data. Experiments conducted on Reuters-21578 and 20NG show that the method is effective especially when the training set is small.

2.2.4 Co-training

In general, a co-training algorithm produces an initial weak classifier from a few labelled examples and later uses unlabelled data to improve its performance. The idea was first introduced in (Blum & Mitchell, 1998). The key defining features of this problem class are that (i) the features can be factored into two (or more) components, i.e. there are two distinct views of an example x, which are redundantly sufficient to correctly classify the example, and (ii) the two components are independent and identically distributed, so that the features in one view of the example x do not always co-occur with the features in the second view. A different approach to co-training is given in (Goldman & Zhou, 2000). See (Abney, 2002; Seeger, 2000) for a comprehensive survey on co-training.

The use of "concepts" derived by clustering as in (Raskutti et al., 2000a) provides an alternate description of the data, similar to the redundant views used in co-training. In this vein, (Raskutti et al., 2002b) present a co-training strategy to make use of unlabelled data. Two predictors are trained in parallel, and each predictor labels the unlabelled data to train the other predictor in the next round. The process repeats for a number of iterations. The predictors are SVMs, one trained using the original word presence features view, and the other trained with solely the new cluster features that are derived by clustering both labelled and unlabelled data. The new features include membership information as well as similarity to clusters' centroids for the more populous clusters as described in their previous work (Raskutti et al., 2000a). This new feature space creates an alternative redundant view of the data as imposed by the co-training framework of (Blum & Mitchell, 1999). They evaluate the method using SVM classifiers on Reuters-21578, 20Newsgroups, and WebKB corpora. Their results are encouraging and confirm previous findings.

A different co-training approach is based on co-training between clustering and classification (Kyriakopoulou, 2007). Unlike the procedure in (Blum & Mitchell, 1999) it does not require a priori the existence of two distinct properties of the underlying data distribution in order to work. Also, it doesn't use two different supervised learning algorithms that complement each other as in (Goldman & Zhou, 2000). Instead, there is one

original feature space, which is used interchangeably by an unsupervised and a supervised learning algorithm, and each algorithm augments it by propagating its results in the form of corresponding *meta*-features. Specifically, following the procedure in (Kyriakopoulou & Kalamboukis, 2007), at every round of co-training a "hard" clustering algorithm groups the examples of the training and testing sets into *k* clusters. The examples that belong to the same cluster are augmented with a *meta*-feature that denotes membership information to this cluster. Then a separate SVM classifier for each class of the classification task is build from the augmented feature space. Each SVM classifier returns a prediction for each example, which is interpreted as the likelihood that the example belongs to a certain class. The predictions of the underlying classifiers for each example are compared and each example is assigned the label of the class with the highest prediction. The labels information is translated into *meta*-features that are used to augment the feature space and the algorithm iterates. According to experimental findings the combination of clustering with classification in a co-training setting, and the addition of corresponding *meta*-features, are successfully used as an additional source of information about margins. The experimental results on widely used datasets demonstrate the superiority of the approach over SVMs.
Table 2 summarizes the methods presented in this section.

2.3 Clustering in large-scale classification problems

Clustering in large-scale classification problems is used as a down-sampling pre-process to classification, in order to select the most representative training examples according to: i) clustering and information from the resulting hyperplane of a SVM initially trained on cluster representatives, ii) clustering and prior class label information, iii) a combination of cases i and ii, iv) solely clustering results, and v) problem decomposition.

2.3.1 Select most representative training data according to clustering and information from the resulting hyperplane of a SVM initially trained on cluster representatives

In this case, first, the training examples are clustered. Then, cluster representatives (clusters' centroids) are used to train an initial SVM classifier. Next, follows a process that selects the clusters that contain the most representative training examples according to a combination of the clustering and classification results. Usually, this process is called declustering and corresponds to an expansion of the training set according to clustering (i.e. the examples of a cluster are no longer represented by the cluster's centroid; instead all the examples are considered). Lastly, a SVM is trained on the new training set. The following algorithms differ in the selection of the cluster representatives, and the way the clustering and classification results are combined in order to select the clusters that contain the best candidates from the training examples. In concluding, they exploit the distributional properties of the training data, i.e. the natural clustering of the training data, and the overall layout of these clusters relative to the decision boundary of SVMs.

- The ***clustering-based SVM (CB-SVM)*** method (Yu et al., 2003) uses the hierarchical clustering technique named BIRCH (Zhang et al., 1996) to cluster the training examples. The key idea of CB-SVM is to use a hierarchical clustering algorithm to get a finer description of the training data closer to a SVM decision boundary and a coarser description away from it. Let T_p and T_n be the hierarchical trees built from the positive and the negative training examples respectively. Then, a SVM is trained from the centroids of the root nodes (i.e. clusters) of T_p and T_n. According to the solution of the

SVM, the clusters whose centroids are support vectors for the SVM and the clusters that are very close to the support vectors (satisfying a certain distance constraint) are declustered into the finer level using the tree structure. These clusters may introduce new support vectors for the SVM, and are thus accumulated into the training set. A new SVM is constructed from the augmented training set, and the declustering process is repeated until nothing is accumulated, i.e. this selective declustering procedure reaches leafs' level. Experiments show that CB-SVM is scalable for very large data sets while also generating high classification accuracy.

Goal	**Authors**	**Clustering/ Classification method**	**Basic method**
Create a training set from the unlabelled data	(Fung & Mangasarian, 2001)	k-means/linear SVM	Unlabelled data selected by k-means are labelled by an oracle or expert.
	(Li et al., 2004)	k-means/linear SVM	Unlabelled data selected by k-means are labelled by cluster labels.
Augment the training set with new documents from the unlabelled data	(Zeng et al., 2003)	k-means/TSVM	Training and unlabelled data are clustered. Unlabelled data nearest to clusters' centroids are added to the training set.
	(Chapelle et al., 2002)	Spectral analysis	Creation of diagonal matrix that contains clustering information.
	(Zhou et al., 2003)		
Augment the dataset with new features	(Raskutti et al., 2000a)	non-hierarchical single-pass clustering algorithm/SVM	Training and unlabelled data are clustered. Each cluster contributes new features to the feature space of the training and testing examples.
	(Takamura, 2003)	hard clustering	The features of the training set are clustered.
	(Kyriakopoulou & Kalamboukis, 2006; Kyriakopoulou & Kalamboukis, 2007)	divisive clustering algorithm /SVM	Training and testing data are clustered. Each cluster contributes a new feature to the feature space of the training and testing examples
Co-training	(Raskutti et al., 2000b)	non-hierarchical single-pass clustering algorithm/SVM	Clustering creates a redundant view in a co-training framework
	(Kyriakopoulou, 2007)	divisive clustering algorithm/SVM	Clustering is used as unsupervised classifier in a co-training framework.

Table 2. Clustering in semi-supervised classification

- (Awad et al., 2004) also apply a hierarchical clustering algorithm, called dynamically growing self-organizing tree (DGSOT) (Luo et al.), as a reduction method of the training set for SVM classification. The authors propose two alternatives to train a SVM for two classes based on the combination of DGSOT and SVM. The first approach generates two hierarchical trees, one for each class, up to a certain level, i.e. they are not fully grown. Then, a SVM is trained on the clusters' references of the trees top nodes (clusters). After computing the margin, the nodes that contain a support vector are declustered by adding their children nodes to the training set. The process of training and declustering is repeated until a stopping criterion holds. In the second approach, one more step is added to the previous procedure before declustering. Specifically, the distance between nodes in the training set is measured. Since the distance between nodes lying in the decision boundary area is the least, the nodes having distance more than the average are excluded. Unlike the approach of (Yu et al., 2003), that first builds the hierarchical tree and then starts to train the SVM, in this approach clustering goes in parallel with training the SVM. During the tree construction and declustering process, DGSOT re-distributes data among newly added children of a node and re-evaluates clustering results. The growth of the tree is controlled, because non-support vector nodes will be stopped from growing, and only support vector nodes will be allowed to grow. Experiments on several datasets against other relevant techniques give contradictory results. The second approach outperforms the rest but needs more time. Also, the algorithm is sensitive on the initial small training set, giving high error rates at the beginning of the training process, which is not fully recovered till the end.
- ***ClusterSVM*** (Boley & Cao, 2004) partitions the training data into pair-wise disjoint clusters using adaptive clustering. Then, a SVM is trained using the centroids of these clusters. Based on this initial SVM, it can be judged whether a cluster contains either only support vectors or only non-support vectors. The clusters that contain both support vectors and non-support vectors based on the decision boundary of the initial SVM are repeatedly divided into sub-clusters that approximately contain either only non-support vectors or only support vectors. Clusters having only non-support vectors are replaced by their representatives. Experiments on artificial and real world datasets prove the efficiency of clusterSVM over popular algorithms such as SMO.
- A similar approach named ***support cluster machines (SCMs)*** (Yuan et al., 2006) uses *k*-means to partition the negative training examples into disjoint clusters, and then trains an initial SVM using the positive examples and the representatives of the negative clusters. With the global picture of the initial SVM, it can approximately identify the support vectors and non-support vectors. A shrinking technique is then used to remove the examples, which are most probably not support vectors. This procedure of clustering and shrinking is performed iteratively until some stopping criteria are satisfied.
- The ***kernel based incremental clustering algorithm (KBIC)*** method uses a scalable kernel based clustering algorithm for the selective sampling based training of non-linear SVMs (Asharaf et al., 2007). This is a two-phase algorithm. In the first phase, KBIC is used to generate a high level description of the data (clusters) in an appropriate kernel induced feature space. The cluster prototypes obtained are used to train a SVM and the corresponding support vectors are identified. In the second phase, a declustering process that expands all the clusters near the boundary creates the training set for the subsequent training of a SVM.

2.3.2 Select most representative training data according to clustering and prior class label information

In this case, the selection of the representative training examples is determined by the composition of the clusters according to the available class label information.

- (Almedia et al., 2000) group the training data in *k* clusters using *k*-means. Clusters formed only by examples that belong to the same class label are disregard and only cluster centres are used. On the other hand, clusters with examples belonging to more than one class labels are unchanged and all training examples are considered. Clusters with mixed composition are likely to happen near the separation margins and they may hold some support vectors. Consequently, the number of training examples for the SVM training is smaller and the training time can be decreased without compromising the generalization capability of the SVM.
- (Fang et al., 2002) apply a clustering approach based on principal component analysis named principle direction divisive partitioning (PDDP) to cluster the training examples. The goal is to minimize noise effects in the training procedure by using those examples that are part of pure clusters, i.e. the ones that are dominated by one of the categories. The training examples that are clustered in pure nodes are used to seed a Naïve Bayes classifier. The authors evaluate the performance of the methods against several interesting variants and show improvements on classification performance.
- (Awad et al., 2004) apply the DGSOT hierarchical clustering algorithm to generate a hierarchical clustering tree from the training examples, and determine the most qualified nodes to decluster based on the heterogeneity of nodes. Heterogeneous nodes are those nodes that have data points assigned to them from different classes, thus, they are more likely to lie in the marginal area between two classes. Then, a SVM is trained on the training examples of the declustered nodes. Experiments on several datasets against other relevant techniques did not give satisfactory results.
- (Cervantes et al., 2006) apply SVM classification based on fuzzy partitioning clustering. The original training set is fuzzy clustered into *k* clusters with respect to a given criterion. The clusters obtained have elements of mixed category or uniform category. SVM is trained on the centroids of the clusters with mixed category elements, because these elements have bigger likelihood to be support vectors. Getting the clusters closer to the decision hyperplane and eliminating the clusters far away reduces the original data set. Then a de-clustering is applied to the reduced clusters and subsets from the original data set are obtained. Finally, SVM is used again and finishes classification. The experimental results show that the number of support vectors obtained using the SVM classification based on the fuzzy partitioning is similar to the normal SVM approach while training time is significantly smaller. However, the number of the clusters k is user-defined in order to avoid computational cost for determining the optimal number of clusters.
- (Li et al, 2007) propose the ***support cluster machine algorithm (SCM)*** to effectively deal with large-scale classification problems. It is a classification model built for clustering. Based on the learning framework of SVMs it defines clustering as a dual optimisation problem with a decision function formulised following the same steps as in SVMs. The goal is to maximize the margin between the positive and the negative clusters of a class, i.e. between clusters obtained only from the positive examples of a class and clusters obtained only from the negative examples accordingly. The examples are clustered using the threshold order-dependent (TOD) clustering algorithm (Friedman & Kandel, 1999). After clustering (or training phase), the training support clusters obtained can be directly used in the decision function to measure the similarity between a cluster and a testing example. The experimental results confirm that the SCM is very effective for

large-scale classification problems due to significantly reduced computational costs for both training and testing and comparable classification accuracies.

2.3.3 Select most representative training data according to clustering, information from the resulting hyperplane of a SVM initially trained on cluster representatives, and prior class label information

This case combines the two previous cases.

- ***Minimum enclosing ball clustering (MEB)*** (Cervantes et al., 2008) employs the concept of core-sets (Badoiu et al., 2002)(Kumar et al., 2003) over the training examples, *L*. The obtained clusters are of the following type: (i) clusters with only positive training examples, Ω^+, (ii) clusters with only negative training examples, Ω^-, and (iii) clusters with both positive and negative examples (or mix-labelled), Ω_m. MEB is used as a data selection method. To this end, only the centres of the Ω^+ and Ω^- clusters and all the examples of the mix-labelled Ω_m clusters are selected to form a reduced training set, L_r, used to train a SVM classifier with the sequential minimal optimisation (SMO) algorithm (Platt, 1998). Then, a de-clustering process augments L_r by including the examples in the clusters whose centres are support vectors of the classifier's solution. Taking the recovered data as new training data set, SVM classification with SMO algorithm is used again to get the final decision hyperplane. The experimental results show that the accuracy obtained by the approach is very close to the classic SVM method, while the training time is significantly shorter, enabling it to successfully handle huge data sets.

2.3.4 Select most representative training data according to solely clustering results

Various assumptions about the clustering results and the information they carry are adopted in order to build the redundant training set.

- (Sun et al., 2004) use k-means to cluster the input space. Because the data that decisively affect SVM classifiers are those at boundary of each class, it is assumed that the data residing on the boundaries of the clusters are critical data that together with the centroid of each cluster are used to train a SVM.
- (Wang et al., 2005) also combine the *k*-means clustering technique with SVM to build classifiers. *K*-means runs on the original training data and all cluster centres are regarded as the compressed data for building classifiers. Accordingly, SVM classifiers are built on the compressed data. The experiments show that it is possible for the algorithm to build classifiers with many fewer support vectors and higher response speed than SVM classifiers. Moreover, testing accuracy of the resulting classifiers can be guaranteed to some extent. This method also employs a parameter tuning method to achieve the required generalization performance at acceptable response time.

2.3.5 Problem decomposition

There are several decomposition methods that try to modify the SVM algorithm so that it can be applied to large datasets.

- The ***clustering support vector machines model (CSVMs model)*** (He at al., 2006) is different from the previous algorithms in this section in that all the training examples are kept during the training process. Using the theory of granularity computing the CSVMs model is able to divide a complex problem into a series of smaller and computationally simpler problems. To accomplish this a *k*-means clustering algorithm is used to cluster the training set into sub-clusters upon which SVMs are subsequently trained in parallel.

Table 3 summarizes the results from this section.

Goal	Authors	**Clustering method**	**Training sample selected or removed**
Select most representative training data according to clustering and information from the resulting hyperplane of a SVM initially trained on cluster representatives (1)	(Yu et al., 2003)	BIRCH	The clusters whose centroids are support vectors for the SVM and the clusters that are very close to the support vectors are declustered
	(Awad et al., 2004)	dynamically growing self-organizing tree	i) Clusters containing support vectors are declustered ii) Distant clusters are removed
	(Boley & Cao, 2004)	adaptive clustering	Clusters having only non-support vectors are replaced by their representatives
	(Yuan et al., 2006)	k-means	Clusters having only non-support vectors are removed
	(Asharaf et al., 2007)	kernel based incremental clustering	Clusters near the boundaries are declustered
Select most representative training data according to clustering and prior class label information (2)	(Almedia et al., 2000)	k-means	Clusters formed by examples that belong to the same class label are disregard and only cluster centres are used. All training examples from clusters of mixed composition are considered.
	(Fang et al., 2002)	principle direction divisive partitioning	Clusters formed by examples that belong to the same class label are considered.
	(Cervantes et al., 2006)	fuzzy partitioning clustering	Clusters of mixed class label composition are declustered and all training examples are considered.
	(Awad et al., 2004)	dynamically growing self-organizing tree	
	(Li et al, 2007)	TOD clustering algorithm	Support clusters obtained in the training phase are directly used in the decision function
Select most representative training data according to (1) and (2)	(Cervantes et al., 2008)	minimum enclosing ball clustering	The centroids of clusters with only positive or only negative training examples, all the examples of clusters with mixed composition, all the examples of the clusters whose centroids are support vectors are used.
Select most representative training data according to solely clustering results	(Sun et al., 2004)	k-means	The centroids and the training data residing at the boundaries of the clusters are selected.
	(Wang et al., 2005)	k-means	The centroids of the clusters are selected.
Problem decomposition	(He at al., 2006)	clustering support vector machines model	All training examples are used. The training set is clustered into subclusters upon which SVMs are subsequently trained in parallel.

Table 3. Clustering in large-scale classification problems

3. Conclusions and future directions

We presented several clustering methods for dimensionality reduction to improve text classification. Experiments show that one-way clustering is more effective than feature selection, especially at lower number of features. Also, when dimensionality is reduced by as much as two orders of magnitude the resulting classification accuracy is similar to a full-feature classifier. In some cases of small training sets and noisy features, feature clustering can actually increase classification accuracy. In the case of IB, various heuristics can be applied in order to obtain finer clusters, greedy agglomerative hard clustering (Slonim & Tishby, 1999), or a sequential K-means like algorithm (Slonim et al., 2002). Co-clustering methods are superior to one-way clustering methods as shown through corresponding experiments (Takamura, 2003). Benefits of using one-way clustering and co-clustering as a feature compression and/or extraction method include: useful semantic feature clusters, higher classification accuracy (via noise reduction), and smaller classification models. The second two reasons are shared with feature selection, and thus clustering can be seen as an alternative or a complement to feature selection, although it does not actually remove any features. Clustering is better at reducing the number of redundant features, whereas feature selection is better at removing detrimental, noisy features. The reduced dimensionality allows the use of more complex algorithms, and reduces computational burden. However, it is necessary to experimentally evaluate the trade-off between soft and hard clustering. While soft clustering increases the classification model size, it is not clear how it affects classification accuracy. Other directions for exploration include feature weighting and combination of feature selection and clustering strategies.

There are four cases of semi-supervised classification using clustering considered in the area. In the first case, in the absence of a labelled set, clustering is used to create one by selecting unlabelled data from a pool of available unlabelled data. In the second case, it is used to augment an existing labelled set with new documents from the unlabelled data. In the third case, the dataset is augmented with new features derived from clustering labelled and unlabelled data. In the last case, clustering is used under a co-training framework. The algorithms presented demonstrate effective use of unlabelled data and significant improvements in classification performance especially when the size of the labelled set is small. In most experiments, the unlabelled data come from the same information source as the training and testing sets. Since the feature distribution of the unlabelled data is crucial to the success of the method, an area of future research is the effect of the source and nature of information in the unlabelled dataset and clustering.

Lastly, clustering reduces the training time of the SVM i) by modifying the SVM algorithm so that it can be applied to large data sets, and ii) by finding and using for training only the most qualified training examples of a large data set and disqualifying unimportant ones. A clustering algorithm and a classifier cooperate and act interchangeably and complementary. In the first case, many algorithms have been proposed (sequential minimal optimisation, projected conjugate gradient, neural networks amongst others) in order to simplify the training process of SVM, usually by breaking down the problem into smaller sub-problems easier to solve. In the second case, the training set is clustered in order to select the most representative examples to train a classifier instead of using the whole training set. The clustering results are used differently by the various approaches, i.e. the selection of the representative training examples follows different methods. Some of the proposed algorithms manage to decrease the number of training examples without compromising the

generalization capability of the SVM. However, there were other approaches that gave contradictory results revealing the difficulty of the problem under examination.
Some methods are applied only on linear problems. Even though some of them can also be used to train non-linear SVMs, the iterative nature of their cluster generation/exploration strategy makes them very expensive to be used in large-scale datasets. There is a need for methods that perform a small number of data scans in order to work. Incremental clustering can also come in useful. Constructing effective indexing structures for non-linear kernels is an interesting direction of future work since it has high practical value especially for pattern recognition of large data sets. Developing an effective indexing structure for high dimensional problems is an interesting direction of future work.
Another important topic for exploration is the choice of the number of word and/or document clusters to be used for the classification task. This and various other parameters are usually defined using various heuristics or are tuned manually. An investigation of automatic approaches to tune the parameters is also desirable.
This review reveals that the area under research is vivid and that clustering is applied in many sub-domains of the problem of text classification. The clustering field can, and indeed must play an important role in enabling effective classification. It is important to invent new designs that are able to support new forms of collaboration but it is essential that this should be done only on the basis of a better understanding of what needs to be accomplished. In this paper, an attempt has been made to achieve such an understanding by abstracting patterns of current applications of clustering to aid classification. We believe that text classification aided by clustering is worthy area of focus for information retrieval, machine learning and artificial intelligence research; both for its direct application and for the insight it gives into other similar problems. Research should focus on model selection and theoretic analysis.

4. References

Abney, S., (2002). Bootstraping. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 360-367.

Almeida, M. B., Braga, A. P., Braga, J. P., (2000). SVM-KM: speeding SVMs learning with a pnori cluster Selection and k-means. *IEEE 6th Brazilian Symposium on Neural Networks*, SBRN 2WO.

Asharaf, S., Murty, M. N., Shevade, S. K., (2007). Cluster based training for scaling non-linear Support Vector Machines. *Proceedings of the International Conference on Computing: Theory and Applications* (ICCTA'07).

Awad, M., Khan, L., Bastani, F, Yen, I. L., (2004). An effective support vector machine SVMs performance using hierarchical clustering, *in Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04)*, pp. 663-667.

Badoiu, M., Har-Peled, S., Indyk. P., (2002). Approximate clustering via core-sets, *in Proceedings of the 34th Symposium on Theory of Computing.*

Baker L. D., McCallum A. K., (1998). Distributional clustering of words for text classification, *Proceedings of SIGIR'98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 96–103, Melbourne, AU. ACM Press, New York, US.

Bekkerman R., El-Yaniv R., Tishby N., Winter Y., (2001). On Feature Distributional Clustering for Text Categorization. *Proceedings of SIGIR'01, 24th ACM International*

Conference on Research and Development in Information Retrieval, pages 146–153, New Orleans, US, ACM Press, New York, US.

Bekkerman R., El-Yaniv R., Tishby, N., Winter Y., (2003). Distributional Word Clusters vs. Words for Text Categorization, *Journal of Machine Learning Research*, 3, 1183-1208.

Blum, A., Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *COLT: Proceedings of the Workshop on Computational Learning Theory*.

Boley, D., Cao, D., (2004). Training support vector machine using adaptive clustering. *In Proceeding of 2004 SIAM International Conference on Data Mining*.

Burges, C. J. C., (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery* 2:121 – 167.

Cervantes, J., Li, X., Yu, W., (2006). Support vector machine classification based on fuzzy clustering for large data sets, *in MICAI 2006 Advances in Artificial Intelligence, Lecture Notes in Computer Science (LNCS)*, vol. 4293, Springer, Berlin, pp. 572-582.

Cervantes, J., Li, X., Yu, W., Li, K., (2008). Support vector machine classification for large data sets via minimum enclosing ball clustering. *Neurocomputing*, Vol. 71, Issue 4-6, pp. 611-619.

Chapelle, O., Weston, J., Scholkopf, B., (2002). Cluster kernels for semi-supervised learning. In *NIPS*, volume 15.

Dai, W., Xue G.R., Yang, Q., Yu, Y., (2007). Co-clustering based classificaiton for out-of-domain documents. *In Proceedngs of KDD 2007*.

Dhillon I., Mallela S., Kumar R., (2002). Enhanced word clustering for hierarchical text classification, *in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, Edmonton, Alberta, Canada, pp. 191-200.

Dhillon I., Mallela S., Kumar R., (2003a). A Divisive Information-Theoretic Feature Clustering Algorithm for Text Classification, *Journal of Machine Learning Research 3*, 1265-1287.

Dhillon I., Mallela S., Modha, S., (2003b). Information theoretic co-clustering. *In Proceedings of the ACM SIGKDD Conference*.

Fang, Y., C., Parthasarathy, S., Schwartz, F., (2002). Using Clustering to Boost Text Classification. *In Proceedings of the IEEE ICDM Workshop on Text Mining*.

Felici, G., Truemper, K., (2002). A minsat approach for learning in logic domains. *Informs. J. Computing*, Vol. 14, No. 1.

Friedman, M., Kandel, A. (1999). Introduction to pattern recognition, *Chapter Distance Functions*, 70–73. London, UK: Imperial College Press.

Fung, G. and Mangasarian, O.L., (2001). Semi-supervised support vector machines for unlabeled data classification. Optim. Methods Software. v15 i1. 29-44.

Goldman, S., Zhou, Y. (2000). Enhancing supervised learning with unlabeled data. *Proc. 17th International Conf. on Machine Learning*, pp. 327–334, Morgan Kaufmann, San Francisco, CA.

He, J., Zhong, W., Harrison, R., Tai, P. C., Pan, Y., (2006). Clustering support vector machines and its application to local protein tertiary structure prediction. *ICCS 2006*, part II,LNCS 3992, pp. 710-717.

Jain, A. K., Dubes, R.C., (1988). Algorithms for Clustering Data. *PrenticeHall*, Englewood Clis, New Jersey.

Koller, D., Sahami, M., (1996). Toward optimal feature selection. *In proceedings of te 13th International Conference on Machine Learning (ICML-96).*

Kumar, P., Mitchell, J.S.B., Yildirim, A., (2003). Approximate minimum enclosing balls in high dimensions using core-sets, *ACM J. Exp. Algorithmics*, 8.

Kyriakopoulou, A., (2007). Using Clustering and Co-training to Boost Classification Performance. *In proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, volume 2, pp. 325-330.

Kyriakopoulou, A., Kalamboukis, T., (2006) Text classification using clustering. *In Proceedings of the ECML-PKDD Discovery Challenge Workshop*.

Kyriakopoulou, A., Kalamboukis, T., (2007). Using clustering to enhance text classification. *In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval,* pp. 805 – 806

Lavelli, A., Sebastiani, F., Zanoli, R., (2004). Distributional term representations: an experimental comparison. *CIKM 2004*: 615-624.

Lewis, D. D., (1992). An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task. *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Li, B., Chi, M., Fan, J., Xue, X., (2007). Support Cluster Machine. *In Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR.

Li, M., Cheng, Y., Zhao, H., (2004). Unlabeled data classification via support vector machine and k-means clustering. *In Proceedings of the International Conference on Computer Graphics, Imaging and Visualization*, CGIV'04, pp. 183-186.

Lin, J., (1991). Divergence measures based on shannon entropy. *IEEE Transactions on Information Theory*, 37 (14):145–51.

Luo, F., Khan, L., Bastani, F., Yen, I.L., Zhou, J., A Dynamical Growing Self-Organizing Tree (DGSOT) for Hierarchical Clustering Gene Expression Profiles, *The Bioinformatics Journal*, Oxford University Press, UK.

Mubaid, H.A., Umair, S.A., (2006). A new text categorization technique using distributional clustering and learning logic. *IEEE Transactions on Knowledge and Data Engineering,* Vol. 18, No. 9.

Ng, A. Y., Jordan, M. I., Weiss, Y, (2002). On spectral clustering: Analysis and an algorithm. In T. Dietterich, S. Becker and Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems (NIPS)*, 14.

Pereira F., Tishby N., Lee L., (1993). Distributional clustering of English words, *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, p. 183-190.

Platt, J., (1998). Fast training of support vector machine using sequential minimal optimization. *In Advances in Kernel Methods: Support Vector Machine*, MIT Press, Cambridge, MA.

Raskutti, B., Ferrá, H., Kowalczyk, A., (2002a). Using Unlabelled Data for Text Classification through Addition of Cluster Parameters. *Proceedings of the Nineteenth International Conference on Machine Learning*, Pages: 514 – 521.

Raskutti, B., Ferrá, H., Kowalczyk, A., (2002b). Combining clustering and co-training to enhance text classification using unlabelled data, *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.*

Rissanen, J, (1989) Stochastic Complexity in Statistical Enquiry. *World Scientific.*

Salton, G. and McGill, M. J. (1983). Introduction to Modern Information Retrieval. *McGraw-Hill*, New York, NY.

Seeger, M. (2000). Input-dependent regularization of conditional density models. *Technical Report*. http://www.kyb.tuebingen.mpg.de/bs/people/seeger/

Slonim, N., Tishby, N., (1999). Agglomerative Information Bottleneck. *Advances in Neural Information Processing Systems*, p. 617-623.

Slonim, N., Tishby, N., (2000). Document Clustering using Word Clusters via the Information Bottleneck Method. *In Proceedings of the ACM SIGIR*.

Slonim, N., Tishby, N., (2001). The power of word clustering for text classification. *In Proceedings of the European Colloquium on IR Research, ECIR*.

Slonim, N., Friedman, N., Tishby, N., (2001). Agglomerative Multivariate Information Bottleneck. *Neural Information Processing Systems (NIPS 01)*.

Slonim, N., Friedman, N., Tishby, N., (2002). Unsupervised document classification using sequential information maximization. In Proc. of SIGIR, pages 129-136.

Spielman, D. Teng, S. (1996). Spectral partitioning works: planar graphs and finite element meshes. *In 37th Annual Symposium on Foundations of Computer Science*. Burlington, VT, pp. 96–105. Los Alamitos, CA: IEEE Comput. Soc. Press.

Sun, S., Tseng, C. L., Chen, Y. H., Chuang, S. C., Fu, H. C. (2004). Cluster-based support vector machines in text-independent speaker identification. *In Proceedings of the International Joint Conference on Neural Network*.

Takamura, H., (2003). Clustering approaches to text categorization, *Doctor's thesis*, NAIST-IS-DT0061014.

Takamura, H., Matsumoto, Y., (2002). Two-dimensional Clustering for Text Categorization. *In Proceedings of Sixth Conference on Natural Language Learning (CoNLL-2002)*, Taipei, Taiwan, pages 29-35, August-September.

Tishby, N. Z., Pereira, F., Bialek, W., (1999). The Information Bottleneck Method. *In Proceedings of the 37th Allerton Conference on Communication, Control and Computing.*

Verbeek, J., (2000a). An information theoretic approach to finding word groups for text classification. *Master 's thesis, Institute for Logic, Language and Computation (ILLC-MoL-2000-03)*, Amsterdam, The Netherlands.

Verbeek, J. (2000b). Supervised Feature Extraction for Text Categorization. *Benelearn: Annual Machine Learning Conference of Belgium and the Netherlands*.

Wang, J., Wu, X., Zhang, C. (2005). Support vector machines based on K-means clustering for real-time business intelligence systems. *Int. J. Business Intelligence and Data Mining*, Vol. 1, No. 1, pp.54–64.

Yaniv R. E., Souroujon O., (2001). Iterative Double Clustering for Unsupervised and Semi-supervised Learning. *In proceedings of the 12th European Conference on Machine Learning, ECML.*

Yu, H., Yang, J., Han, J., (2003). Classifying large data sets using SVMs with hierarchical clusters, *in Proceedings of the 9th ACM SIGKDD 2003*, Washington, DC, USA.

Yuan, J., Li, J., & Zhang, B. (2006). Learning concepts from large scale imbalanced data sets using support cluster machines. *Proceedings of the ACM International Conference on Multimedia*, (pp. 441–450).

Zhang, T., Ramakrishnan, R., Livny, M. (1996). BIRCH: An efficient data clustering method for very large databases. *Proc. of the 1996 ACM SIGMOD Int'l Conf. on Management of Data*, pp. 103–114.

Zeng, H., J., Wang, X., H., Chen, Z., Lu, H., Ma, W., Y., (2003). CBC: Clustering Based Text Classification Requiring Minimal Labeled Data. *In Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM'03).*

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., Scholkopf, B.. (2003). Learning with local and global consistency. *In 18th Annual Conf. on Neural Information Processing Systems.*

A Review of Past and Future Trends in Perceptual Anchoring.

Silvia Coradeschi and Amy Loutfi
Örebro University
Sweden

1. Introduction

Anchoring is the problem of how to create, and to maintain in time the connection between the symbol- and the signal-level representations of the same physical object. In particular robotic systems with symbolic components need to solve the anchoring problem in order to connect the information present in symbolic form with the sensor data that the robot obtains from the physical world. Previously, solutions to the anchoring problem have been implemented on a system by system basis and could therefore only be applied to restricted domains. However, in recent years, the study of anchoring problem per se has gained an increased interest and an attempt to frame the anchoring problem and provide a theoretical groundwork to dealing with anchoring particularly on artificial systems have been explored. The first definition of anchoring was in (Coradeschi & Saffiotti, 2000) and a community working on anchoring has been established in a number of workshops and a special journal issue (Coradeschi & Saffiotti, 2001;2003;2004). In this chapter, we present the latest developments in anchoring and outline the future trends. In addition, a specific framework is outlined and used as an example to illustrate the main challenges to be addressed in perceptual anchoring.
The anchoring problem is concerned with the grounding of symbols that refer to specific object entities such as "a cup" or even more specifically "cup-22". Anchoring is not concerned with the process of grounding general properties such as "blue" or general concepts such as "difficult". This is the symbol grounding problem (Harnad, 1990) and anchoring is rather a subset of symbol grounding that is limited to physical objects. Anchoring must take the flow of continuously changing sensor input into account to allow for object persistence in time and space. Even though some properties of an object may change while others remain static, the symbol-percept correspondence should remain intact and should hold the current and updated information. This dynamic maintenance of information differentiates anchoring from pattern recognition which for the most part does not take into account this dynamic aspect or the presence of symbols. One way to consider persistence is to have an internal structure which reifies the correspondence between symbols and sensor data. Thus, many of the contributions in anchoring focus on this internal representation and its formalism.

2. Perceptual anchoring in robotics

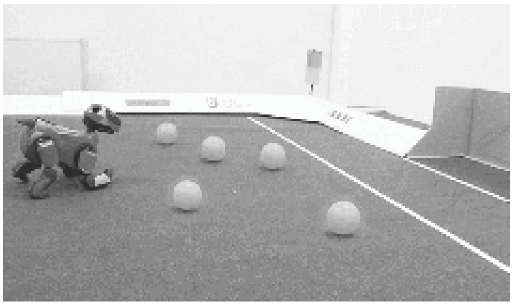


Fig. 1. Examples of anchoring in robotics. (Left) A Robocup domain where similar objects create ambiguities. (Right) Human robot interaction in a home environment where symbolic references to objects are commonly used (photograph by courtesy of Federico Pecora).

Traditionally, anchoring can be seen as a process which creates a shared representation to link several subsystems of an agent, such as the planner to the motion control. In bottom-up approaches, the sensor data determines the initiation of an anchoring process, whereas top-down approaches may initiate an anchoring process upon request. In robotic systems, a number of key challenges are relevant for both bottom-up and top-down anchoring processes. First, uncertainty and ambiguity arise when dealing with real sensors. For these reasons, anchoring may need to consider a number of sub-processes or functionalities which can handle uncertainties and also recover if incorrect decisions are taken. In addition, symbolic descriptions eventually used to link to the perceptual data can be vague and cannot be assessed in terms of a specific quantified sensor value. For instance the concept "a large ball" where the concept large can refer to a range of values whose boundaries are not well defined (Coradeschi et al, 2001). Ambiguous cases can also occur where perceptually similar objects are equally valid candidates in the result of a request. In these cases, further actions may be necessary to resolve the ambiguity (Karlsson et al., 2008). Further, to facilitate human robot interaction, symbols are rarely used in isolation but rather as part of a semantic network where ontological and common sense knowledge plays an important role. As a result, symbolic descriptions may be subject to interpretation and need to handle or cope with variances. For example in Fig. 1, an agent can receive a command to "find a ball", or "find the closest ball" using both definite and indefinite types of references. Further, in scenarios where multiple agents are present, it is important to coordinate and achieve consensus among agents so that a common anchoring is possible.

3. An example of an anchoring framework

Here we present an instantiation of an anchoring framework and its core functionalities that illustrate how an anchoring modality works in a real robotic system. Other frameworks have been explored and a discussion of these contributions is given at the end of this section.

The anchoring framework here is based upon (Coradeschi & Saffiotti, 2000) and contains the following main ingredients:

- A *symbol system* including: a set **X** = {x_1, x_2, …} of individual symbols (variables and constants); a set **P** = {p_1, p_2, …} of predicate symbols; and an inference mechanism whose details are not relevant here.

- A *perceptual system* including: a set $\Pi = \{\pi_1, \pi_2, \ldots\}$ of percepts; a set $\Phi = \{\varphi_1, \varphi_2, \ldots\}$ of attributes; and perceptual routines. A percept is a structured collection of measurements assumed to originate from the same physical object; an attribute φ_i is a measurable property of percepts, with values in the domain $\mathbf{D}_i$.
- A *predicate grounding relation* $g = \mathbf{P} \times \Phi \times \mathbf{D}$, that embodies the correspondence between unary predicates and values of measurable attributes.

The perceptual system generates percepts and associates each percept with the observed values of a set of measurable attributes and the symbol-percept correspondence is reified in an internal data structure, called an *anchor*. Since new percepts are generated continuously within the perceptual system, this correspondence is indexed by time.

At every moment t, $\alpha(t)$ contains: a symbol, meant to denote an object; a percept, generated by observing that object; and a signature, a collection of property values meant to provide the (best) estimate of the values of the observable properties of the object.

To handle anchors, we require functionalities able to create, maintain and remove anchors.

3.1 Creation of anchors

The creation of anchors can occur in both a top-down and bottom-up fashion. Bottom-up acquisition is driven by an event originating from a sensing resource (e.g. the recognition of a segmented region in an image) when perceptual information which cannot be associated to any existing anchor is perceived. Top-down acquisition occurs when a symbol needs to be anchored to a percept, such a call may originate from an external user or a top-level module (e.g. planner).

Acquire

This functionality initiates a new anchor whenever a percept is received which currently does not match any existing anchor. It takes a percept π, and return an anchor α , defined at t and undefined elsewhere. To make this problem tractable, a priori information is given with regards to which percepts to consider. In bottom-up acquisition, a randomly generated symbol is attributed to the anchor. Furthermore, information about the object and its properties are included into the world model used by the planner, in this way the object can be reasoned about and acted upon.

Find

Takes a symbol x and a symbolic description and returns an anchor α defined at t (and possibly undefined elsewhere). It checks if existing anchors that have already been created by the **Acquire** satisfy the symbolic description, and in that case, it selects one. Otherwise, it performs a similar check with existing percepts (in case, the description does not satisfy the constraint of percepts considered by the **Acquire**). If a matching percept is found an anchor is created. Matching of an anchor or percept can be either partial or complete. It is partial if all the observed properties in the percept or anchor match the description, but there are some properties in the description that have not been observed.

3.2 Maintenance of anchors

At each perceptual cycle, when new perceptual information is received, it is important to determine if the new perceptual information should be associated to existing anchors. The following functionality addresses the problem of tracking objects over time.

Track

The track functionality takes an anchor α, defined for t-k and extends its definition to t. The track assures that the percept pointed to by the anchor is the most recent and adequate perceptual representation of the object. We consider that the signatures can be updated as well as replaced but by preserving the anchor structure we affirm the persistence of the object so that it can be used even when the object is out of view. This facilitates the maintenance of information while the robot is moving as well as maintaining a longer term and stable representation of the world on a symbolic level without catering to perceptual glitches.

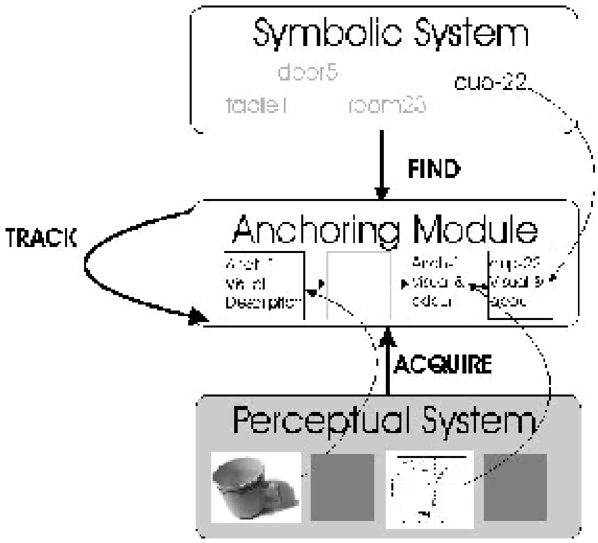


Fig. 2. Graphical illustration of the anchoring functionalities where bottom-up and top-down information is possible and different sensing modalities are used (Louti et. al, 2005).

3.3 Deletion of anchors

By having an anchor structure maintained over time, it is possible to preserve the perceptual information even if the object is not currently perceived (caused by the object being out of view and/or by the inaccuracy in the measurement of perceptual data). The challenge is to determine if the association of new percepts is justified or whether certain anchors should be removed. Mechanisms for destroying invalid anchors need to be in place. This is a difficult problem, because conceptually it is not clear when it is appropriate to remove anchors from the system. Anchors could be removed if they are not relevant for the current task, because the object to which it refers has been physically removed from the environment or the reliability of the perceptual information has expired. Anchors may also need to be removed if they have been associated to invalid perceptual data such as sensory glitches. We currently adopt simple solutions in which objects that are not perceived when expected decrease in a "life" value of the respective anchor. When the anchor has no remaining life, the anchor is removed. The decreasing life of anchors is shown in Fig. 4. A more adequate strategy to handle the maintenance of anchors may also be to include a "long term" memory where anchors may be stored for future use.

3.4 Integration of the functionalities

The event-based functionalities are restricted to the **Find** and **Acquire** while the **Track** functionality is regularly called. Fig. 4 shows an overview and an example of the framework and its functionalities. In the example in the figure, anchors are created bottom-up from the visual percepts. Later, additional features of that object are required, for example, the olfactory property. These features are stored in the anchor. When a top-down request is sent to the anchor module to find a cup with matching properties denoted by the symbol "cup-22", the **Find** functionality anchors the symbol to the perceptual data.

As seen in the figure, properties can be collected at different time points using different modalities. Even when certain perceptual properties are updated, such as the smell property, which may change over time, other perceptual properties are maintained. Conversely, if the visual percepts of an anchor are replaced, the smell property previously obtained is not lost. In this way, the anchor is used to compensate for any dynamically changing features of an object. Furthermore, the perceptual description of anchors can be accessed by the planner to reason about perceptual knowledge. In certain cases, this may result in specific calls to perceptual actions in order to disambiguate between similar objects.

3.5 Case study

Here we outline a brief example of how the anchoring module operates within a simple corridor monitoring scenario. In each corridor there may be several objects, in this case garbage cans. The robot automatically toggles between the task of patrolling the corridor, inspecting objects and waiting for commands from the user. Patrolling the corridor involves moving from corridor to corridor in a discovery for new objects and recognition of previous objects. When an inspect is invoked, the robot visits each object collecting the odour property. The inspect is usually autonomously invoked when new objects are detected.

The robot is equipped with several heterogeneous sensing modalities such as a camera, sonar, tactile sensors, and an electronic nose. In this example, the modalities of interest are the vision, and e-nose. The vision component is trained to detect any visual signal matching garbage cans. For each found object we extract a number of properties such as color, size and relative position using different heuristics. The collection of the properties belonging to an object is called a percept.

The electronic nose component is able to classify odours providing a symbolic categorical description (Loutfi et al., 2005). The robot is also able to localize itself within the corridor using odometry and has a number of high level processes such as a planner which reasons about actions and a plan executor which can monitor motion control.

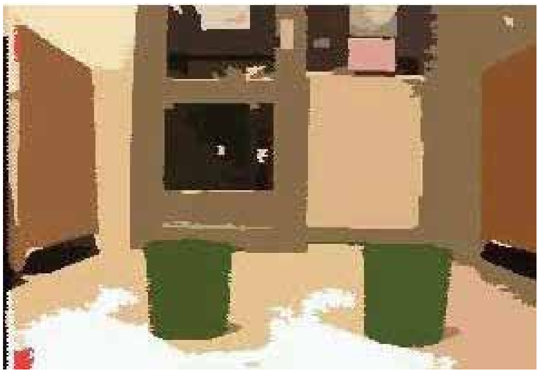
Before we begin to outline the corridor example, let us first examine the structure of an anchor. Fig. 3 shows two anchor structures that have been created bottom-up from a segmented image. The anchoring module updates anchors such that at every moment t, $\alpha(t)$, contains:

Name - For top-down anchors a name is a symbol denoting the object in the planner (e.g., Silvia's Cup). For anchors that have been created bottom-up the name is initially arbitrary.

Symbolic description In general a symbolic description is given in a top-down fashion, however for bottom-up anchors the symbolic description may also be derived directly from the perceptual information of the object. For example, the odour classification module may populate the symbolic description with the linguistic odour name when classifying an odour associated to a particular object.

Perceptual description The perceptual description is a vector which consists of the important properties of the object such as position (relative and global), colour, shape, and when available the odour signature.

In the figure the two anchors, *Gar-36* and *Gar-34*, are visually similar however *Gar-36* currently has an olfactory property. In the current implementation an anchor is "baptised" with the name of the percept which initially invoked the bottom-up process. As will be shown in the next example, the percept may be updated but the anchor persists using the tracking functionality.



```
(ANCHOR
:NAME GAR-34
:ID ANCHOR-6
:SYMBOLIC-DESCRIPTION ((SHAPE = GARBAGE
CAN) (COLOUR=GREEN) )
:PERCEPTUAL-DESCRIPTION (<TR GAR-35 30 [KN
29 GREEN #1#GARBAGE L<1958,347>~0
BL<1958,347>~0 :IN-ROOM CORR-2 :NEC 0.21000004
:LIFE 0.8]>
```

```
(ANCHOR
:NAME GAR-36
:ID ANCHOR-8
:SYMBOLIC-DESCRIPTION ((SHAPE = GARBAGE)
(COLOUR=GREEN)(SMELL=ETHANOL))
:PERCEPTUAL-DESCRIPTION (<TR GAR-36 30 [KN
29 GREEN #1=GARBAGE L<1979,-173>~0 BL<1979,-
173>~0 :IN-ROOM CORR-2 :NEC 0.21000004 :LIFE
0.8]>)
```

Fig. 3. (Left) Segmented image from the vision module observing two green garbage cans at the center of the screen. (Right) The anchors created in a bottom-up manner for each object. *Gar-34* refers to the garbage can on the left of the image and *Gar-36* refers to the garbage can on the right.

In Fig. 4 The local space of the robot together with the visual image from the camera as well as the creation, deletion and updating of anchors is depicted during a run through the corridor. The figure contains four snapshots described as follows:

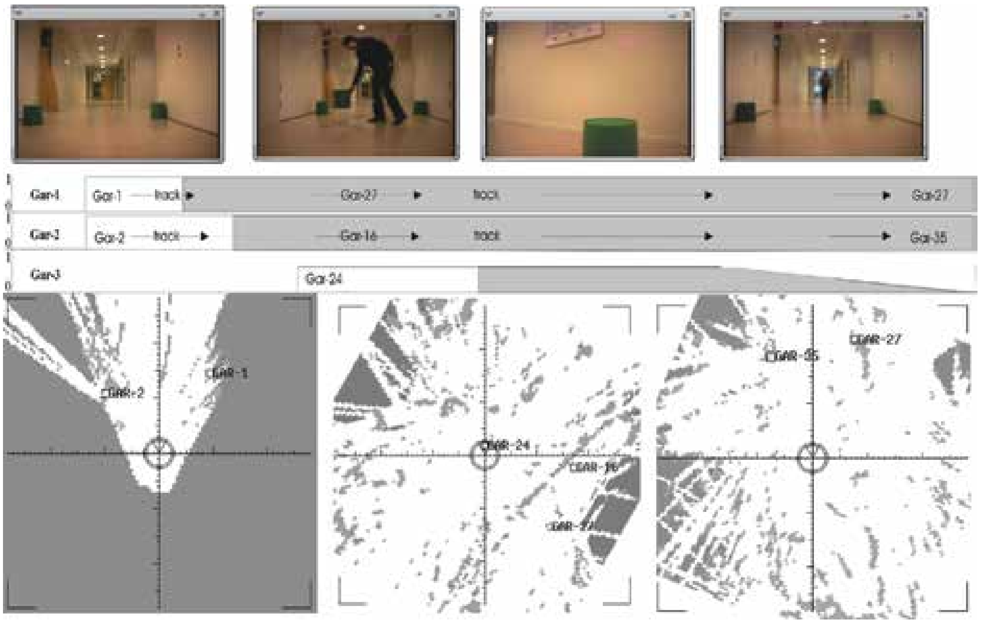


Fig. 4. The top row shows the camera images at different time points, the middle row shows the activity at the anchoring level. Grey bars indicate anchors with olfactory properties. The bottom row shows the corresponding local perceptual space given the changing representation of visual percepts (Loutfi et. al, 2005).

Scene 1 - The robot begins patrolling the corridor, two visual percepts are detected and two anchors denoted by *Gar-1* and *Gar-2*, are created. An inspect is performed and both anchors obtain olfactory properties, shown in the Figure by the grey colouring. Since the anchors are created in a bottom-up fashion their labels are arbitrary.
Scene 2 - As the robot continues its patrol, another object is inserted into the environment at a later time. Note however, that the previous two anchors are still maintained by the track functionality. Although the local space shows only the current percepts, the anchoring module updates the link between the anchor *Gar-1* and the percept *Gar-27*. A new anchor is also created for the third object denoted by *Gar-3* with visual percept *Gar-24*.
Scene 3 - The robot approaches the object in order to acquire its odour property and the result is stored in the corresponding anchor. Some time later, the object is removed from the environment. The life of the anchor slowly decreases when an expected percept is no longer detected.
Scene 4 - The anchor is removed from the system and unless it is perceived again, its properties cannot be accessed by the find functionalities described above.
This simple scenario shows how the anchoring module is used to create an internal structure which can then maintain the perceptual coherence of objects, considering each object has both spatial and olfactory properties. Even when visual properties of anchors are being updated, the stored smell property remains until a new odour character is acquired by the next inspect action. The previous odour character is then stored in the odour repository.

3.6 Other approaches to anchoring

The example above illustrates the main theoretical ingredients necessary for an anchoring module. In the literature, the study of anchoring per se has led to different approaches to address the problem of maintenance and creation of the symbol-percept correspondence referring to objects. Chella et. al, (2003) present a framework where conceptual spaces (Gärdenfors, 2000) are used to combine in a unary formalism all features referring to a specific object, and consequently the combination of the features referring to the object is a single point inside the conceptual space. Similarly, Bonarini et al (2001) have also presented an anchoring framework where concept layer is used to combine features while also using previously established domain knowledge, from a "world modeller". Modayil & Kuipers (2007) examines unsupervised learning approaches to bootstrap an ontology of objects to sensor input from a robot. Four multiple learning stages are combined in which an object is first individualized, then tracked and described (using shape models) and finally categorized. A collection of works have also extended the anchoring framework beyond the traditional notion of physical objects and contends with: embodied interactions between the robot and objects in the its environment (Chinellato et al, 2007), human movement (Fritsch et al, 2003), actions sequences represented in situation calculus to dynamic properties of objects using conceptual spaces (Chella et al. 2007), perceptually indistinguishable objects (Santore & Shapiro, 2004).

4. Cooperative anchoring

In the previous sections, anchoring has only been considered in the context of single robotic systems. In the case of multiple robotic systems with different and heterogeneous devices cooperating, the anchoring problem undertakes a new complexity. In a distributed system, individual agents may need to anchor objects from perceptual data coming either from sensors embedded directly on the robot or information coming from other devices. Further, agents each with its own anchoring module may need to reach a consensus in order to successfully perform a task.

Sensor fusion plays an important role for multi-agent or "cooperative" anchoring. A cooperative anchoring approach based on the presented framework has been explored in (LeBlanc & Saffiotti, 2008) which considers primarily the problem of fusing pieces of information coming from a distributed system. In this work, both complex devices such as mobile robots and simple devices contain pieces of information which may need to be fused together in order to create a global notion of an anchor. Each agent maps items of information into its own anchor space (inspired by Gardenfors' conceptual spaces) where an anchor space is a multi-dimensional domain such as colour, position, weight etc. The individual anchor spaces are mapped into a shared anchor space and from within the shared space information is compared and combined as needed. This is done using the fuzzy intersection of n-dimensional fuzzy sets of the individual anchor spaces. Fig. 5 shows a concrete example where a block is seen from two cameras and an RFID is acquired by an RFID reader. The information is fused in a shared anchoring space using fuzzy sets.

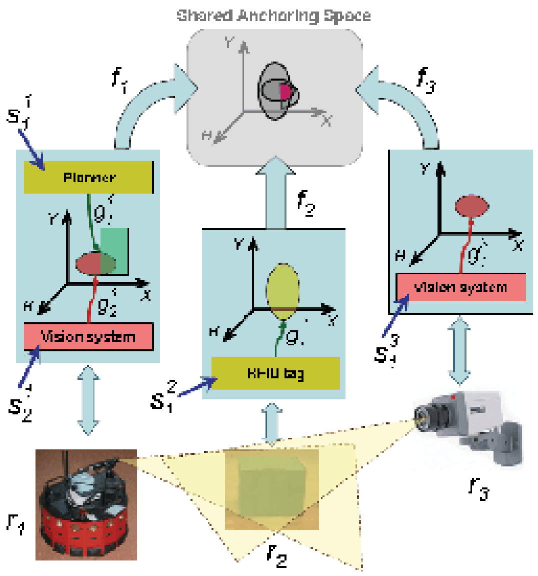
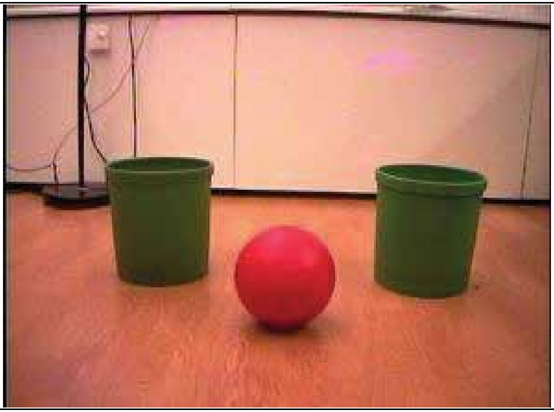


Fig 5. Different elements in a scenario where a mobile robot and a camera mounted in a ceiling detect a parcel and perform cooperative anchoring (courtesy of K. Leblanc, Leblanc & Saffiotti, 2008).

Another proposed solution for dealing with multi-robot anchoring also extends single—robot systems presented in the previous section. Bonarini et al, (2007) extend their framework to a multi-agent case by combining the information from different agents in a global representation at the conceptual level using a fusion model based on clustering techniques.
Decentralized approaches have also been considered in (Guirnaldo et al, 2004) where each agent has its own anchoring module and broadcasts its anchors to other agents. In this approach, agents have defined roles of leaders and followers and in case of conflict the leader's anchor is accepted, thus it is not clear how fusion would occur if two equally ranked agents conflict. The challenge of achieving an agreement among agents about the objects that are perceived is an open problem. The challenge of reaching an agreement has been studied in the multi-agent community in (Goldman et al, 2007; Kararzyniak & Pieczynska, 2006). Such work can form the basis for a system where agreement or consensus can be achieved between multiple agents using decentralized anchoring.

5. Anchoring for human robot interaction

Another emerging trend is to study the anchoring process that occurs together with human operators and users. Anchoring is specially suited to HRI application since the symbolic level has clear benefit while communicating with non-experts. Communicate about objects is often central in HRI and such communication requires a coordinated symbol-percept link between human and robot.



```
? (FIND-ANCHOR 'ANCH
'((SHAPE = GARBAGE) (COLOR = GREEN)))
- FOUND 2 CANDIDATES: PLEASE CHOOSE
- 1. GREEN GARBAGE LEFT BEHIND OF
RED BALL
- 2. GREEN GARBAGE RIGHT BEHIND OF
RED BALL
? 1
- REFORMULATING:
- (FIND-ANCHOR 'ANCH '((SHAPE =
GARBAGE) (COLOR = GREEN) (LEFT-OF =
BALL-2) (BEHIND-OF = BALL-2)))
- FOUND: ((ANCHOR ANCH-1 ANCH ...))
```

Fig 6. Spatial Relations used to resolve ambiguous situations.

A dialogue system for human-robot collaborations is a particular instance of the anchoring problem, when dialogue about physical objects is concerned. An example of such a dialogue system is explored in (Kruijff & Brenner, 2007), there information about the object state as well as a history of the object state is used to describe changes in a scene. An important feature of this approach is that it considers descriptions that contain spatial relations among the objects. Spatial relations are crucial when human describe and recognize objects. While communication among devices can be based on coordinates this is not meaningful when the communication is with humans. Further works on using spatial relations and computation of spatial relations between anchors for human robot interaction was explored in (Melchert et. al, 2007). In this work, the spatial relations were used to provide meaningful object descriptions but also could facilitate human participation in the anchoring by using human interaction in the disambiguation process between visually similar anchors. In Fig. 6, an example is shown where a request to find a green garbage can is sent to the anchoring system. The anchoring system cannot disambiguate between the two identical garbage cans and ask the user if he means garbage can on the left and behind the red ball or the one on the right. The user selects the first option and a new request containing the additional information is sent to the anchoring module that succeeds in finding the object. The returned descriptions for the spatial relations of objects present all possible relations of objects. For cases of HRI it would be more beneficial to generate object descriptions with salient and relevant information for the human users (Jordan & Walker, 2005) .
Other works which examine human participation in the anchoring process include (Yu & Ballard, 2004). Here a learning approach is used where spoken names of objects are grounded to image data representing the object. Similarly (Roy, 2005) explores a theoretical framework for involving human participation in the grounding of language to both perception and action using a manipulator robot.

6. Anchoring in symbiotic robotic system

Symbiotic robotic systems are an emerging trend in robotics that combine many of the ingredients in the previous two sections, namely many devices operating in parallel and human users interacting with the system (Coradeschi & Saffiotti, 2006). The advantage of the symbiotic robotic systems is that many of the current challenges in robots can be circumvented, for instance localization can be helped by cameras on the ceiling and an id of an object can be provided by en RFID tag. However, a symbiotic system require a solution of the two anchoring problems just mentioned, that is, cooperative anchoring and anchoring in cooperation with humans with the additional difficulty that the solutions should be compatible and guarantee a coherent anchoring process. Consider as an example the following scenario:

> "Johanna, an elderly woman living alone in her apartment, has a medical condition which affects her blood pressure. Suddenly, while cooking, Johanna feels faint and must sit down. She signals to Emil, her domestic robot, and asks where she last left her blood pressure medicine. Emil communicates with other devices in the home and a camera in the bedroom detects a small bottle on the bedside table. To recognize whether this is the correct medicine, the vacuum cleaner robot already present in the bedroom, is sent to the bedside table. The bottle is successfully recognized as Orvaten (used to treat Johanna's hypotension) using the RFID reader on the vacuum cleaner robot. Emil tells Johanna that the medicine is on the bedside table and Johanna then asks Emil to fetch the medicine for her."

In this scenario, the information from the camera and the RFID reader needs to be combined to recognize the correct medicine and an anchor needs to be established that connects "the blood pressure medicine of Johanna" with the sensor data corresponding to the object and coming from the different devices. The position of the object is also stored and can be then used by Emil to get the medicine. An important challenge in symbiotic systems is the establishment of a shared ontology where concepts referring to objects are coherent between agents, robots, pervasive devices and most importantly the human users. Such ontology forms the basis of the communication among the participant in the anchoring process and provides additional information that can be used in the anchoring process such as function of objects, how different part of objects are related and classes and subclasses of objects. For example, Orvaten is used to treat hypotension, is inside a bottle with an etiquette, and is a subclass of medication. Generate object descriptions that are both meaningful to a specific agent and salient to a task is also essential in systems where different actors are present. In the scenario, the most useful description to Johanna is that the medicine is on the bedside table; while Emil who fetches the medicine may need the actual color and shape of the bottle.

The study of anchoring within the symbiotic system has been examined in a few cases. In Mastrogiovanni et al. (2007) a symbolic data fusion system for an ambient intelligent environment is presented consisting of several cognitive agents with different capabilities. Lopes et al. (2002) describe a way to utilize the KRR component for knowledge acquisition and information disambiguation. Similarly, in (Melchert et al. 2007) we have also examined how KRR system such as LOOM can be integrated into an anchoring framework in the context of the symbiotic system for improved cooperation between devices and human users.

7. Conclusions

For artificial intelligence to be used as tools for robotic systems, it is important to be able to capitalize on the work in symbolic AI systems. To accomplish this goal, it is necessary to connect the symbolic information to the sensory percepts from the robotic system. This chapter has discussed this important aspect especially concerning the symbol-percept correspondence referring to physical objects. This problem has been defined as the anchoring problem and a number of examples of anchoring in practice have been given. Furthermore, three emerging trends for anchoring has been highlighted: cooperative anchoring, anchoring for HRI and anchoring in symbiotic robotic system where greater symbolic processing is used and thus creating additional challenges for anchoring.

8. References

A. Bonarini, M. Matteucci, and M. Restelli. Anchoring: do we need new solutions to an old problem or do we have old solutions for a new problem? *In Proc. AAAI Fall Symposium on Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems*, 2001.

A. Bonarini, M. Matteucci, and M. Restelli. Problems and solutions for anchoring in multi-robot applications. *Journal of Intelligent and Fuzzy Systems*, 18:245–254, 2007.

A. Chella, M. Frixione, and S. Gaglio. Anchoring symbols on conceptual spaces: the case of dynamic scenarios. *Robotics and Autonomous Systems*, 43(2):175-188(14), 2003.

A. Chella, H. Dindo, and I. Infantino. Imitation learning and anchoring through conceptual spaces. *Applied Artificial Intelligence*, 21(4&5):343–359, 2007.

E. Chinellato, A. Morales, E. Cervera, and A. Del Pobil. Symbol grounding through robotic manipulation in cognitive systems. *Robotics and Autonomous Systems*, 55(12):851–859, 2007.

S. Coradeschi, D. Driankov, L. Karlsson, and A. Saffiotti. Fuzzy anchoring. In Proc of the *IEEE Intl Conf on Fuzzy Systems*, pages 111–114, Melbourne, AU, 2001.

S. Coradeschi and A. Saffiotti. Anchoring symbols to sensor data: preliminary report. In Proc. of the 17th *American Association for Artificial Intelligence Conf.* (AAAI), pages 129–135, 2000.

S. Coradeschi and A. Saffiotti, editors. Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems: Papers from the AAAI Fall Symposium. AAAI Press, Menlo Park, California, 2001.

S. Coradeschi and A. Saffiotti. An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43(2-3):85–96, 2003.

S. Coradeschi and A. Saffiotti, editors. *Robotics and Autonomous Systems, special issue on Perceptual Anchoring*. Elsevier Science, 2003.

S. Coradeschi and A. Saffiotti, editors. Anchoring symbols to sensor data. Papers from the AAAI Workshop Technical Report WS-04-03. AAAI Press, Menlo Park, California, 2004.

S. Coradeschi and A. Saffiotti. Symbiotic robotic systems: Humans, robots, and smart environments. *IEEE Intelligent Systems*, 21(3):82–84, 2006.

G. Cortellessa, A. Loutfi, and F. Pecora. An on-going evaluation of domestic robots. In *Proc. of the HRI-08 Workshop on Robotic Helpers*, pages 87–91, Amsterdam, NL, 2008.

J. Fritsch, M. Kleinehagenbrock, S. Lang, F. Loemker, G. A. Fink, and G. Sagerer. Multi-modal anchoring for human-robot-interaction. *Robotics and Autonomous Systems*, 43(2):133-147(15): 2003.

P. Gärdenfors. *Conceptual Spaces: The Geometry of Thought*. MIT Press, Cambridge,MA, 2000.

C. Goldman, M. Allen, and S. Zilberstein. Learning to communicate in a decentralized environment. *Autonomous Agents and Multi-Agent Systems*, 15(1):47–90, 2007.

S. Guirnaldo, K.Watanabe, and K. Izumi. Enhancing the awareness of decentralized cooperative mobile robots through active perceptual anchoring. *International Journal of Control, Automation and Systems*, 2:450–462, 2004.

S. Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

P. Jordan and M. Walker. Learning content selection rules for generating object descriptions in dialogue. *Journal Artif. Intell. Res. (JAIR)*, 24:157–194, 2005.

L. Karlsson, A. Bouguerra, M. Broxvall, S. Coradeschi, and A. Saffiotti. To secure an anchor - a recovery planning approach to ambiguity in perceptual anchoring. *AI Communications*, 21(1):1–14, 2008.

R. Katarzyniak and A. Pieczynska. The outline of the strategy for solving knowledge inconsistencies in a process of agents' opinions integration. In *6th International Conference Computational Science*, Volume 3993 of Lecture Notes in Computer Science, pages 891–894, 2006.

G. Kruijff and M. Brenner. Modelling spatio-temporal comprehension in situated human-robot dialogue as reasoning about intentions and plans. In *Symposium on Intentions in Intelligent Systems*, AAAI Spring Symposium Series, 2007.

K. LeBlanc and A. Saffiotti. Cooperative anchoring in heterogeneous multi-robot systems. In *Proc. of the IEEE Int. Conf. on Robotics and Automation* (ICRA), Pasadena, CA, 2008.

L.S. Lopes. Carl: from situated activity to language level interaction and learning. In Proc. Intl. Conf. on *Intelligent Robots and Systems*, pages 890–896, Lausanne, 2002.

A. Loutfi, M. Broxvall, S. Coradeschi, and L. Karlsson. Object recognition: A new application for smelling robots. *Robotics and Autonomous Systems*, 52:272–289, 2005.

A. Loutfi, S. Coradeschi and A. Saffiotti. Maintaining Coherent Perceptual Information Using Anchoring. *In Proc. of the Nineteenth International Joint Conference on Artificial Intelligence*. 2005.

F. Mastrogiovanni, A. Sgorbissa, and R. Zaccaria. A distributed architecture for symbolic data fusion. In Proceedings of 20th *International Joint Conference on Artificial Intelligence*, Hyderabad, India, 2007.

J. Melchert, S. Coradeschi, and A. Loutfi. Knowledge representation and reasoning for Perceptual anchoring. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI),* Patras, Greece, 2007.

J. Melchert, S. Coradeschi, and A. Loutfi. Spatial relations for perceptual anchoring. In *Proceedings of AISB'07, AISB Annual Convention,* Newcastle upon Tyne, UK, 2007.

J. Modayil and B. Kuipers. Autonomous development of a grounded object ontology by a learning robot. *National Conference on Artificial Intelligence (AAAI-07)*, 2007.

D. Roy. Semiotic Schemas: A Framework for Grounding Language in the Action and Perception. *Artificial Intelligence*, 167(1-2): 170-205, 2005.

J. Santore and S. Shapiro. Identifying an object that is perceptually indistinguishable from one previously perceived. *In Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pages 968–969. 2004.

C. Yu and D. Ballard. On the integration of grounding language and learning objects. *In Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pages 488–494, 2004.

16

A Cognitive Vision Approach to Image Segmentation

Vincent Martin and Monique Thonnat
INRIA Sophia Antipolis Méditerranée, project-team PULSAR
France

1. Introduction

Image segmentation consists in grouping pixels sharing some common characteristics. In vision systems, the segmentation layer typically precedes the semantic analysis of an image. Thus, to be useful for higher-level tasks, segmentation must be adapted to the goal, i.e. able to effectively segment objects of interest. Our objective is to propose a cognitive vision approach to the image and video segmentation problem. More precisely, we aim at introducing learning and adaptability capacities into the segmentation task. Traditionally, explicit knowledge is used to set up this task in vision systems. This knowledge is mainly composed of image processing programs (e.g., specialized segmentation algorithms and post-processing's) and of program usage knowledge to control segmentation (e.g., algorithm selection and algorithm parameter settings).

In real world applications, when the context changes, so does the appearance of the images. It can be due to local changes (e.g., shadows, reflections) and/or global illumination changes (e.g., due to meteorological conditions). The consequences on segmentation results can be dramatic. This context adaptation issue emphasizes the need of automatic adaptation capabilities. Our first objective is to learn the contextual variations of images in order to discriminate between different segmentation actions. The identification of the contexts will lead to different segmentation actions as algorithm selection.

When designing a segmentation algorithm, internal parameters (e.g., thresholds or minimal sizes of regions) are set with default values by the algorithm authors. In practice, it is often up to an image processing expert to supervise the tuning of these free parameters to get meaningful results. As seen in Figure 1, it is not clear how to choose the best parameter set regarding the segmented images: the first one is quite good but several parts of the insect are missing; the second one is also good, since the insect is well outlined, but too many meaningless regions are also present. However, complex interactions between free parameters make the behaviour of the algorithm fairly impossible to predict. Moreover, this awkward task is tedious and time-consuming. Thus, the algorithm parameter tuning is a real challenge. To solve this issue, our objective is threefold: first, we want to automate this task in order to alleviate users' effort and prevent subjective results. Second, the fitness function used to assess segmentation quality should be generic (i.e. not application dependent). Third, no *a priori* knowledge of segmentation algorithm behaviours is required, only ground truth data should be provided by users.

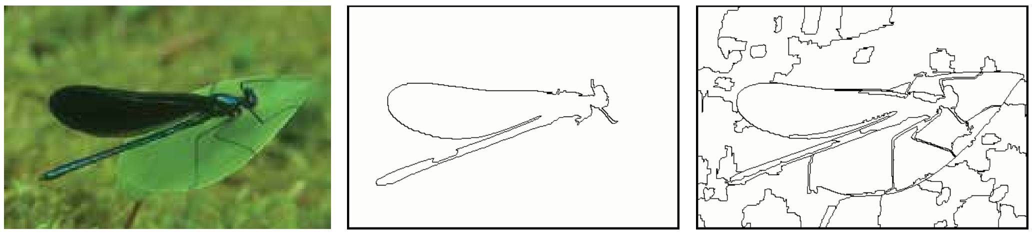


Fig. 1. Illustration of the problem of algorithm parameter tuning. An image is segmented with the same algorithm (based on colour homogeneity) tuned with two different parameter sets.

The very first problem of segmentation is that a unique general method still does not exist: depending on the application, algorithm performances vary. This is illustrated in Figure 2 where two different algorithms are applied on the same image. The first one seems to be visually more efficient to separate the ladybird from the leaf. The second one produces too many regions not very meaningful. Basically, two popular approaches exist to set up the image segmentation task in a vision system. A first approach is to develop a new segmentation algorithm dedicated to the application task. A second approach is to empirically choose an existing algorithm, for instance by a trial-and-error procedure. The first approach leads to develop an *ad hoc* algorithm, from scratch, and for each new application. The second approach does not guarantee adapted results and robustness. So, a need exists for developing a new approach to the algorithm selection issue. When facing different algorithms, this approach should be able to automatically choose the one best suited with a segmentation goal and the image content.

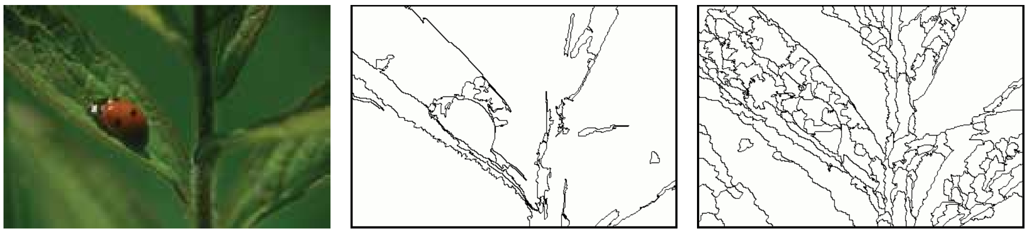


Fig. 2. An example of the segmentation of an image with two different algorithms. The first algorithm forms regions according to a multi-scale colour criterion while the second uses a local colour homogeneity criterion.

Once all the algorithms have been optimized, a third issue is to select the best one. However, when images of the application domain are highly variable, it remains quite impossible to achieve a good segmentation with only one tuned algorithm. Our objective is to make use of the extracted knowledge of context variations and parameter tuning to associate a segmentation action to each identified context.

Finally, in many computer vision systems at the detection layer, the goal is to separate the object(s) of interest from the image background. When objects of interest and/or image background are complex (e.g. composed of several subparts), a low-level algorithm cannot achieve a semantic segmentation, even if optimized. For this reason, a fourth issue is to refine the image segmentation to provide a semantically meaningful segmentation to higher vision modules.

Our final objective in this chapter is to show the potential of our approach through a segmentation task in a real-world application. The segmentation task we focus on is image segmentation in a biological application related to early pest detection and counting. This implies to robustly segment the objects of interest (mature white flies) from the complex background (rose leaves). Our goal is to demonstrate that the cognitive vision system coupled with our adaptive segmentation approach achieves a better detection rate of white flies than tuned with an *ad hoc* segmentation.
This chapter is structured as follows. Section 2 introduces the reader to image segmentation in the context of computer vision systems. We propose an overview on topics closely related to our problem. Section 3 details each step of our learning approach. Section 4 shows how the learnt segmentation knowledge is used to perform adaptive image segmentation. The next section is dedicated to the validation of the approach for a real world application: the segmentation step of a cognitive vision system dedicated to the recognition of biological organisms in static images. Concluding remarks and suggestions for future work are discussed in section 6.

2. Related work

In this section, we present some previous work related to image segmentation, segmentation performance evaluation, algorithm parameter optimization and algorithm selection.

2.1 Image segmentation

Several surveys of segmentation techniques have been published. Three of them (Pal & Pal, 1993; Skarbek & Koschan, 1994; Lucchese & Mitra, 2001) review about 300 publications giving a fair overview of the state-of-the-art in segmentation at the image-based processing level. Pal and Pal (Pal & Pal, 1993) mainly evaluate algorithms for grey-valued images and introduce three of the first attempts to exploit colour information. Skarbek and Koschan (Skarbek & Koschan, 1994) concentrate their survey on colour image segmentation. They classify the algorithms according to the underlying concepts of the homogeneity predicate and identify four categories: pixel-based, area-based, edge-based and physics-based approaches. Lucchese and Mitra (Lucchese & Mitra, 2001) also review exclusively colour segmentation approaches and use a similar categorization: feature space based, image domain based and physics based techniques. We can summarize these studies by making some important remarks, closely akin to the conclusions of (Skarbek & Koschan, 1994) in their survey:

1. General purpose algorithms are not robust and usually not algorithmically efficient.
2. All techniques are dependent on parameters, constants and thresholds which are usually fixed on the basis of few experiments. Tuning and adapting parameters is rarely performed.
3. As a rule, authors ignore comparing their novel ideas with existing ones.
4. As a rule, authors do not estimate the algorithmic complexity of their methods.
5. It seems that separating processes for region segmentation and for object recognition is the reason of failure of general purpose segmentation algorithms.
6. Several different colour spaces are employed for image segmentation. Nevertheless, no general advantage of one of the colour spaces with regard to the other colour spaces has been found yet.

2.2 Segmentation performance evaluation

Considering the increasing number of segmentation algorithms, the problem of performance segmentation evaluation becomes a primordial task. Two reasons motivate this statement: researchers must be able to compare their algorithm to other ones, and end-users must be able to choose an algorithm depending on the problem to solve. Usually, segmentation results are visually assessed by the algorithm's designer, which only allows subjective and qualitative conclusions on the algorithm performance. A generic method for the segmentation evaluation task does not exist, but many approaches have been proposed and can be classified into two principal classes: unsupervised methods and supervised methods (see Figure 3). The first class gathers the methods which do not require any *a priori* knowledge of segmentation results to evaluate. Their principle consists in estimating empirical criteria based on image statistics. The second class groups together evaluation methods based on *a priori* knowledge as a reference segmented image, usually named a ground truth (GT). A good survey of all these methods can be found in (Zhang, 1996) and in (Rosenberger et al., 2005).

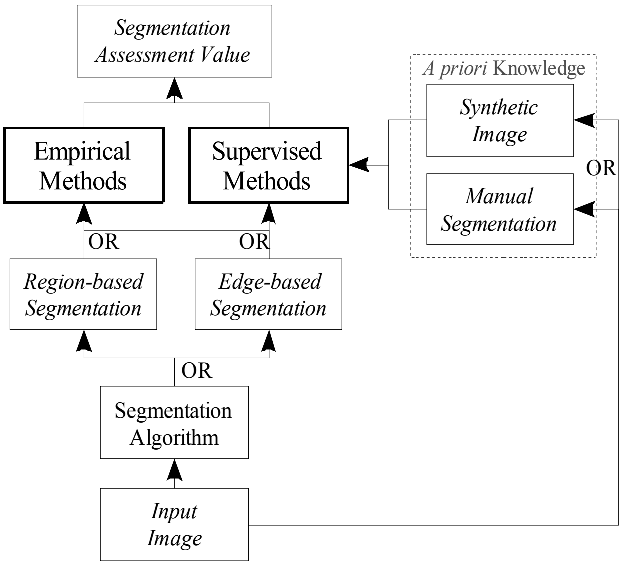


Fig. 3. Segmentation evaluation diagram starting from an input image and returning a segmentation assessment value

2.2.1 Unsupervised methods

The major advantage of unsupervised methods is that they do not require the intervention of an expert, just the definition of a metric of quality/discrepancy measure by the user is needed. Thus, these methods are totally automatic. However, defining a metric that could match all the segmentation objectives defined by the user is not a tricky task. Hence, quality measures are at best heuristic, since no specific knowledge of object(s) to segment is available. This tends to consider unsupervised performance evaluation method not very pertinent. Among the variety of proposed discrepancy measures, we can cite the well-known Rosenfield, Borsotti, Rosenberg or Charbrier criteria. A recent survey of these unsupervised methods can be found in (Zhang, 2008).

2.2.2 Supervised methods

Reference segmentations are achieved generally by hand or by generating synthetic images. In the last case, the ground truth data are objective and precise, in the contrary of subjective and imprecise hand-made expert drawing. These methods try to determine how far the actually segmented image is from the reference image in a quantitative manner, e.g. based on the number of misclassified pixels versus the reference segmentation. There are also a variety of discrepancy methods for the supervised evaluation of image segmentation. Some interesting ones can be found in (Yasnoff et al., 1977), (Everingham et al., 2002), and (Mezaris et al., 2003). The use of a ground truth is double-edged: it makes this class of methods potentially the most general and the less biased but this also supposes that ground truths are easily available. From this study, it also clearly appears that multi-objective methods yield better results than stand-alone methods (edge-based or region-based). However, the manner to combine measures remains an issue.

If we take a look at the number of publications around the segmentation evaluation problem, we can see that at present, this number is about one thousand concerning the segmentation algorithms, one hundred concerning the evaluation methods, and does not raise ten concerning the comparison of evaluation methods. If more efforts have been recently put on segmentation evaluation, it is still difficult to define wide-ranging performance metrics and statistics. Several explanations justify this limitation: (1) no common mathematical model or general strategy for evaluation is available especially for analytic methods; (2) no single evaluation can cover all aspects of segmentation algorithms; (3) appropriate ground truths are hard to determine objectively. Then, to overcome such limitations, potential research directions may explore methods combining multiple metrics in an effective manner (e.g., using learning) and methods considering the final goal of the segmentation.

Research is currently underway in terms of using these metrics as a mean to optimize parameters within a segmentation algorithm or to select the best adapted algorithm. This involves using an optimization procedure which is also a challenge in the context of image segmentation. The next section discusses this issue.

2.3 Algorithm parameter optimization

In this section, we relate some work dealing with segmentation algorithm parameter optimization. All the following approaches rely on three independent components: a segmentation algorithm with its free-parameters to tune, a segmentation quality assessment function and a global optimization algorithm as seen in Figure 4.

Researchers have experienced many segmentation optimization approaches during the last decade. Almost all of the free derivative optimization techniques have been tested. Interesting frameworks can be found in (Bahnu et al., 1995; Peng & Bahnu, 1998; Mao et al., 2000; Cinque et al., 2002; Gelasca et al., 2003; Pignalberi et al., 2003; Abdul-Karim et al., 2005). In the worst case, results of optimized segmentations are equivalent to the ones obtained with default parameters. In most of the cases, segmentation quality is improved and time spent to tune algorithms is drastically reduced. The authors present their frameworks as generic by nature and then widely applicable. This affirmation is well-founded in an analytical point of view since the three main components are considered separately. Nonetheless, each described framework has been set up for a particular segmentation task where the fitness function has been specifically elaborated for the application using implicit domain knowledge. Thereby, it has not been proved how the fitness function can affect the performance of the optimization. Moreover, if authors have often assessed their optimization methods against default segmentations, they did not make any quantitative evaluation regarding to other optimization techniques. A comparative study of optimization algorithms has to be done.

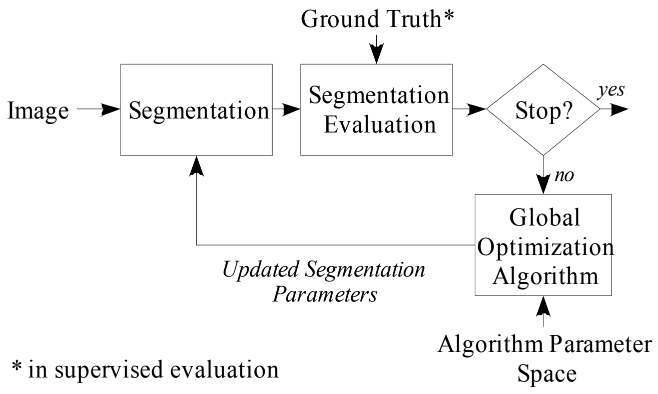

Ground Truth*
Image
Segmentation
Segmentation Evaluation
Stop?
yes
no
Global Optimization Algorithm
Updated Segmentation Parameters
Algorithm Parameter Space
* in supervised evaluation

Fig. 4. The segmentation parameter optimization framework

2.4 Algorithm selection

In this section, we focus on the algorithm selection problem. Here, the goal is not to find the best parameter setting but rather to find the most suitable algorithm among several ones for a given segmentation task. Due to the still increasing number of algorithms, this problem has taken a big interest during the last decade. Basically, researchers tackle the problem with two different philosophies: model representation approach versus expert system approach.
In (Xia et al., 2005), the authors make the assumption that the choice of a segmentation algorithm can be predicted from a global feature vector. In other words, this means that a relationship between algorithm behaviours and global variations of image characteristic can be established (by means of learning techniques). The principal drawback is that the training process is imitated by the user assessment reliability. The task of visual algorithm ranking is time-consuming and then hardly conceivable in the case of large image and algorithm sets. As depicted by the authors, objective performance evaluation criteria (i.e. automatic) should be investigated to free users from the tedious training stage. In (Zhang & Luo, 2000), the authors propose a framework for automatic algorithm selection based on knowledge driven hypothesis-and-test optimization model. An expert system is designed to use evaluation, heuristic, and high-level knowledge (as *a priori* restrictions about domain dependent object features) to segment an image with the best adapted segmentation algorithm.
Globally, the two approaches rely on strong hypothesis concerning their field of applications: variations between images must be easy to model, algorithm behaviours within the images must be well-established, and high-level knowledge of objects to segment must be provided as a key-element of the performance evaluation. Actually, the lack of theory on segmentation rules out these approaches to be universally applicable. Indeed, application domains with image variations difficult to model disable the model representation approach, and the expensive knowledge acquisition task needed to build

expert systems limits their applicability. We can add that the model representation approach appears to be more realistic in a computing point of view as compared to expert systems.

2.5 Conclusion

We have reviewed the segmentation task in the field of computer vision systems. If researchers agree that segmentation is one of the fundamental problems in computer vision, the efforts devoted to cope with this issue since the last four decades have still not led to a unified solution. Most of the vision systems are application dependent and their segmentation step is based on heuristic rules for, as example, the tuning of algorithm parameters. It is, however, well-established that such *a priori* knowledge is determined by domain experts from the context in which the segmentation takes place. Hence, the generalization to other domain of application is strongly limited. Nonetheless, it appears that the recent cognitive vision approach (ECVISION, 2005) has identified some avenues of researches to cope with these limitations, as integration of machine learning techniques into the knowledge acquisition task.

3. Supervised learning for image segmentation

In this section, we present our cognitive vision approach to image segmentation. We have defined in section 1 the expectations of the segmentation task in computer vision systems (context adaptation, algorithm selection and tuning). We have seen in section 2 that these challenging issues have been tackled by many different approaches. Our goal is to propose a methodology that takes the best of each approach.
In the context of cognitive vision, we propose a framework with a reusability property to ease the set up of the segmentation task in vision systems. More precisely, our framework does not require image segmentation skills: the complexity of this tricky task is hidden by means of automatic algorithm parameter tuning and segmentation assessment. Moreover, the acquisition of the segmentation knowledge is made convenient by user-friendly interactivity. The second property of cognitive vision we are aiming at is the property of genericity. In our framework, the different components are not application dependent. Consequently, this framework can be used with different segmentation algorithms and for different real-world applications. The third cognitive property of our framework is its adaptation faculty to image content and to application needs. To this end, we use learning techniques for context adaptation, algorithm selection and parameter tuning.

3.1 Overview

Our framework consists of two stages: a learning stage and an adaptive segmentation stage. The framework relies on training data composed of manual segmentations of the training images with semantic region annotations. The learning stage extracts the segmentation knowledge from the training data by means of:

- a data mining module to extract and learn contextual variations,
- an optimization procedure for automatic segmentation parameter tuning,
- a learning module for context adaptation (i.e. to associate a segmentation action to each identified context),
- a learning module for semantic segmentation; the goal is to train region classifiers with respect to the annotated manual segmentations of the training images.

The learning stage is sketched in Figure 5. The module for adaptive image segmentation relies on the learnt segmentation knowledge. It will be described in section 4. The following sections details each step of the learning stage.

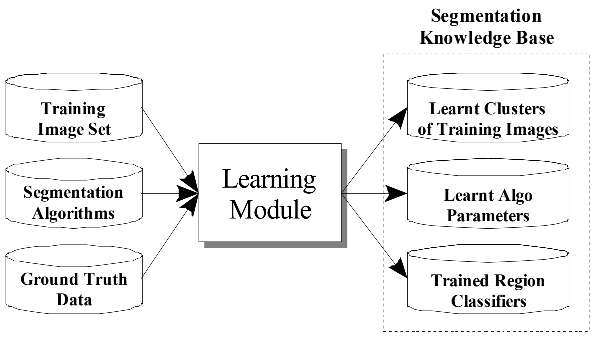


Fig. 5. The learning module for adaptive image segmentation.

3.2 Data mining for learning image contextual variations

Our strategy for algorithm selection is to tackle the problem *a priori* of the segmentation. In this case, the goal is not to directly select the algorithm depending on its relative performance evaluation but depending on the image to segment. Usually, variations between images lead to a variability in the segmentation. As a consequence, similar images should be segmented with the same algorithm and different images should be segmented with different algorithms or different parameter settings. These variations can be induced by changes in background appearance, changes in illumination source, or changes in imagery device configuration. The goal is to identify the different situations leading to different segmentation configurations. To this end, we define the context of an image as the quantitative representation of its local and global characteristics. Practically, the context is described by a d–dimensional feature vector $v(I)$ extracted from the whole image (e.g., a colour histograms). In our experiments, we have used a Density-Based Spatial clustering algorithm called DBScan proposed by Ester et al. (Ester et al., 1996) to identify the image clusters. This algorithm is well-adapted for clustering noisy data of arbitrary shape in high-dimensional space as histograms. Starting from one point, the algorithm searches for similar points in its neighbourhood based on a density criterion to manage noisy data. Non clustered points are considered as 'noise' points. The runtime of the algorithm is of the order $O(n \log n)$ with n the dimension of the input space. DBScan requires only one critical input parameter, the *Eps*-neighbourhood, and supports the user in determining an appropriate value for it. A low value will raises to many small clusters and may also classify a lot of points as noisy points, a high value prevents from noisy point detection but produces few clusters. A good value would be the density of the least dense cluster. But it is very hard to get this information on advance. Normally one does not know the distribution of the points in the space. If no cluster is found, all points are marked as noise. In our approach, we set this parameter so as to have at the most 15% of the training images classified as 'noise' data.
We denote κ a cluster of training images belonging to the same context θ. The set of the n clusters is noted $\mathrm{K} = \{\kappa_1, \ldots, \kappa_n\}$ and the corresponding context set $\Theta = \{\theta_1, \ldots, \theta_n\}$. Once the clustering is done, the internal data structures (here R-trees) and the DBScan parameters (*Eps*-neighbourhood, cluster IDs, etc.) are learnt.

3.3 Learning for segmentation parameter tuning

In this section, we detail our parameter optimization framework. The goal is to optimize the parameterisation of segmentation algorithms according to ground truth segmentations of the training images. For this task, the user must provide:

1. Manual segmentations of the training images with closed outlined regions.
2. Segmentation algorithms with their free parameters, i.e. the sensitive parameters to be tuned, as well as their range values. This kind of knowledge is often given by the algorithm's author.

3.3.1 Formalization of the optimization problem

Let I be an image of the training image set $\Im$, G_I be its ground truth (e.g. a manual segmentation), A be a segmentation algorithm and p^A vector of its free parameters. The segmentation of I with algorithm A is defined as $A(I, p^A)$. We define the segmentation quality E_I^A with the assessment function ρ as follows:

$$E_I^A = \rho\left(A\left(I, p^A\right), G_I\right) \tag{1}$$

The value E_I^A is an assessment value of the matching between the segmentation and the ground truth. This can be goodness or a discrepancy measure.

The purpose of our optimization procedure is to determine a set of parameter values $\hat{p}_I^A$ which minimizes/maximizes:

$$\hat{p}_I^A = \arg\min/\max_{p^A} \rho\left(A\left(I, p^A\right), G_I\right) \tag{2}$$

The final assessment value $\hat{E}_I^A$ and the optimal parameter set $\hat{p}_I^A$ make a pair sample noted $\left(\hat{p}_I^A, \hat{E}_I^A\right)$. This pair forms the segmentation knowledge for the image I and the algorithm A. The set of all collected pairs constitutes the segmentation knowledge S set such that:

$$S = \bigcup_{I \in \Im} \left(\hat{p}_I^A, \hat{E}_I^A\right) \tag{3}$$

One key-point of this optimization procedure is the definition of the assessment function ρ. The quality of the final result varies according to this fitness function. The choice of a segmentation performance evaluation metric is hence fundamental. It is discussed in the next section.

3.3.2 Definition of the segmentation performance evaluation metric

As stated in section 2.2, it is not obvious to select a performance evaluation metric because no single metric can cover all aspects of segmentation algorithms. We propose to use a boundary-based metric and to evaluate the segmentation in terms of both localization accuracy and the shape accuracy of the extracted regions. The biggest advantage of boundary-based metrics against region-based metrics is their lower computational cost. It is always faster to count and compare some boundary pixels than a lot of region pixels. This metric is broadly usable since it mainly relies on generic concepts (false and missed boundary pixel rates).

The region boundary set for the ground truth and for the segmentation result are noted B_I^G and B_I^A respectively. Two types of errors are considered: missing boundary rate e_m^B and false boundary rate e_f^B. The former, e_m^B, specifies the percentage of the points on B_I^G that are mistakenly classified as non-boundary points; while the latter, e_f^B, indicates the percentage of the points in B_I^A that are actually false alarms. Therefore,

$$e_m^B = \frac{|T_1|}{B_I^G} \text{ and } e_f^B = \frac{|T_2|}{B_I^A} \tag{4}$$

where

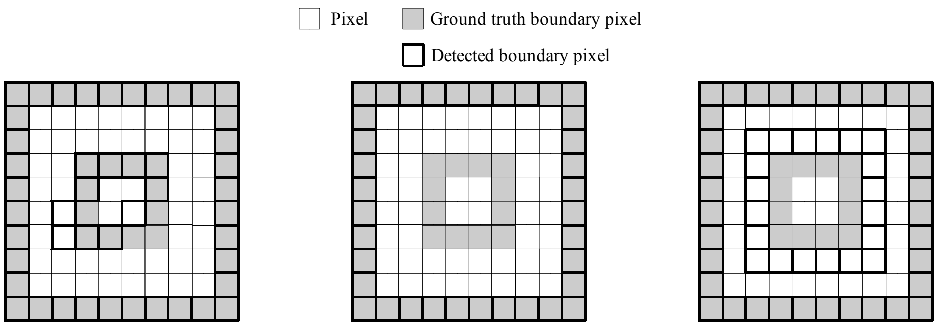
$$\begin{aligned} T_1 &= \{x \mid (x \in B_I^G) \wedge (x \notin B_I^A)\} \\ \text{And } T_2 &= \{x \mid (x \in B_I^A) \wedge (x \notin B_I^G)\} \end{aligned} \tag{5}$$

and |.| is the cardinal operator.
We define the segmentation quality E_I^A with the assessment function as follows:

$$E_I^A = \rho(B_I^A, B_I^G) = \frac{1}{2}(e_m^B + e_f^B) \tag{6}$$

with $E_I^A \in [0,1]$.

The value $E_I^A = 0$ indicates perfect boundary pixel matching between the segmentation result and the ground truth when using algorithm *A*. The value $E_I^A = 1$ indicates that all pixels are misclassified. However, it is easy to show that this metric comes up against unsuited response to under-segmented results, as illustrated in Figure 6. Segmentation in panel (a) shows two regions with a quite good ground truth overlap, only three pixels are misclassified. In the panel (b), the segmentation shows only one region and
the quality score is logically less than in (a). In the last panel (c), two regions are present but the centre region badly overlaps the corresponding ground truth centre region. In opposition with visual assessment, the segmentation quality is worst than in Figure 6(c).



(a) $e_m^B = \frac{3}{48}, e_f^B = \frac{3}{47}$ $E_I^A = 0.063$

(b) $e_m^B = \frac{12}{48}, e_f^B = \frac{0}{36}$ $E_I^A = 0.125$

(c) $e_m^B = \frac{12}{48}, e_f^B = \frac{20}{56}$ $E_I^A = 0.304$

Fig. 6. Limitation of the segmentation evaluation metric when weighting terms (w_m^B and w_f^B) are not used.

The metric is improved by introducing two weighting terms w_m^B and w_f^B which quantify the average distance between misclassified points to the ground truth boundary such that:

$$w_m^B = \frac{1}{|T_1|} \sum_{x \in T_1} dist\left(x, \hat{x}_I^A\right) \tag{7}$$

with $\hat{x}_I^A$ the closest pixel to x belonging to B_I^A, and

$$w_f^B = \frac{1}{|T_2|} \sum_{x \in T_2} dist\left(x, \hat{x}_I^G\right) \tag{8}$$

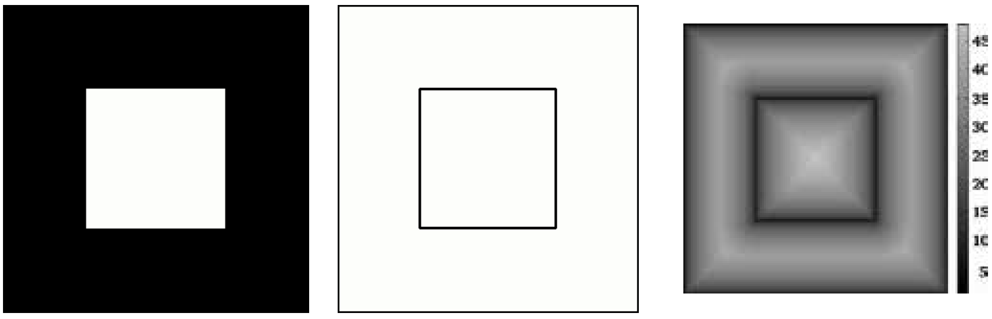
with $\hat{x}_I^G$ the closest pixel to x belonging to B_I^G; $dist(x_1, x_2)$ is the Euclidean distance between two pixels $x_1(u,v)$ and $x_2(u,v)$ in a 4-neighbourhood such that:

$$dist\left(x_1, x_2\right) = \sqrt{\left(x_1(u) - x_2(u)\right)^2 + \left(x_1(v) - x_2(v)\right)^2} \tag{9}$$

Since w_m^B and w_f^B have no fixed upper bounds, the normalization factor is useless and the segmentation quality measure becomes:

$$\begin{aligned} E_I^A &= w_m^B \times e_m^B + w_f^B \times e_f^B \\ &= \frac{1}{B_I^G} \sum_{x \in T_1} dist\left(x, \hat{x}_I^A\right) + \frac{1}{B_I^A} \sum_{x \in T_2} dist\left(x, \hat{x}_I^G\right) \end{aligned} \tag{10}$$

The search of $\hat{x}_I^A$ (resp. $\hat{x}_I^G$) is made easier by the use of a distance map (Maurer & Raghavan, 2003) computed from B_I^A (resp. B_I^G). This operation is exemplified in Figure 7.
By taking back the example in Figure 6 with the new definition of the evaluation metric, the values of E_I^A for the cases (a), (b), and (c) are respectively 0.168, 0.75, and 0.679, yielding a good correlation with a visual assessment.



(a) region-based segmentation composed of 2 regions

(b) region boundary representation of the segmentation in (a)

(c) distance map of (b), the grey-level value of a pixel represents the Euclidean distance to the nearest boundary pixel

Fig. 7. An example of a distance map from a binary contour segmentation.

Once our performance evaluation metric is defined, the goal is now to minimize the segmentation error E_I^A in order to learn optimal segmentation parameters. This is the role of our closed-loop global optimization procedure.

3.3.3 Choice of the optimization algorithm

Of primary importance in this optimization procedure is finding an optimal segmentation parameter setting $\hat{p}_I^A$ for each $I \in \Im$. We also aim at providing a good evaluation study of the tested optimization techniques in terms of performance versus computational cost and parameter setting. In the family of free derivative techniques, we propose the following criteria to assess the optimization algorithms:

1. Since the segmentation of an image is the most expensive process in the optimization loop, the number of maximum segmentation algorithm calls might be set as a parameter. Indeed, even if the ultimate goal of an optimization procedure is to find a global optimum, the computational cost should remain realistic.
2. The optimization algorithm must be able to converge whatever the evaluation profile, i.e. robust enough to find (quasi-)global optimum of various non-smooth functions.
3. The final quality of the optimization procedure should no be too dependent of the tuning of the optimization algorithm parameters, whatever the segmentation algorithm.

We have seen in our survey (see section 2.3) that several optimization techniques have been applied to tackle the segmentation optimization problem. Although all of them are suitable with our problem, no comparative study exists to help us in our choice. Thus, we have decided to focus on two techniques which are worth being compared. The first one is the Simplex algorithm (Nelder & Mead, 1965) and the second is a standard genetic algorithm (Goldberg, 1989) using non-overlapping populations and optional elitism. In one hand, simplex is easy to use, fast to converge, but requires to define a initializing strategy (starting point(s) and starting step) and do not guarantee to find a global optimum. In an other hand, genetic algorithms are robust but are slower to converge and their parameters must be set carefully.

After all pair samples $\left(\hat{p}_I^A, \hat{E}_I^A\right)$ have been extracted for all segmentation algorithms to test, the next step is to select and tune the one(s) which will be learnt for each identified context. The following section discusses our learning strategy for context adaptation.

3.4 Learning for context adaptation

The previous parameter optimization step allows us to objectively compare the segmentation algorithms with regards to their best performance scores. A straightforward strategy for the selection of an algorithm is thus to take the first best. Nevertheless, the problem becomes more difficult when the training images are heterogeneous, due for instance to global or local variations in the background. In this case, one segmentation algorithm could be the best adapted for the segmentation of a training image subset and another one for another subset. We propose to tackle this problem by associating one algorithm per subset. More precisely, we propose to rank the segmentation algorithms for each previously identified context. The context adaptation strategy can be formalized as follows:

$$\begin{aligned} f : R^d &\to S \\ v(I) &\mapsto (A, \hat{p}^A) \end{aligned} \tag{11}$$

However, it is impossible to continuously predict the algorithm behaviour according to image variations and therefore the function cannot be seen as a regression model. Our approach is to tackle this modelling problem by applying an unsupervised clustering of the training images to identify the different contexts, i.e. clusters of images having similar feature vectors. Then, for each cluster (i.e. images of the same context), segmentation algorithms are ranked and the best one is learnt. The best algorithm is the one performing the best average performance on the cluster. For each algorithm, a mean parameter set is computed as follows:

$$\overline{p}^A = \frac{1}{|\Im_A|} \sum_{I \in \Im_A} \hat{p}_I^A \tag{12}$$

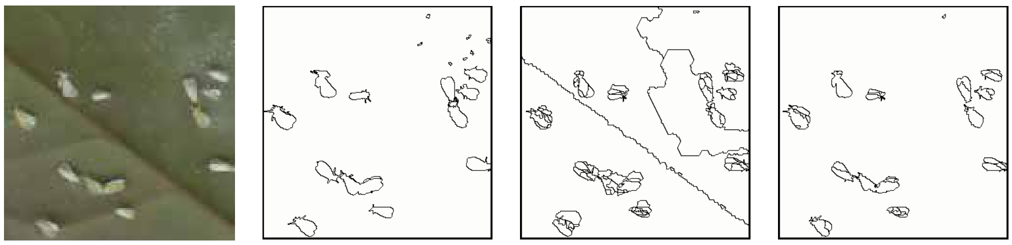
where $\Im_A$ is the subset of training images for which the algorithm A has obtained the best evaluation results among the other algorithms. Finally, for each training image of the cluster and each algorithm A tuned with $\overline{p}^A$, the segmentation quality is computed again. The algorithm having the best average performance on the training image set is finally selected.
We obtain a discrete function F taking a context identifier θ as input and returning an algorithm A with a mean parameter setting $\overline{p}^A$ such as:

$$\begin{aligned} F &: \Theta \to S \\ \theta &\mapsto (A, \overline{p}^A) \end{aligned} \tag{13}$$

This selection strategy comes to select the robustest algorithm based on objective comparisons, i.e. the algorithm which can deliver the best results for the cluster with a globally relevant parameter set. However, this straightforward ranking approach has two major drawbacks. First, by selecting only one algorithm and averaging its parameters, it reduces the previously extracted segmentation knowledge amount to one mean case. Second, even if the selected algorithm over performs the others in most of the cases, the parameter averaging can have disastrous effects on the algorithm performance.
The principal purpose of this strategy is to overcome the drawbacks of a pure global ranking strategy by dividing the solution space and by restricting the ranking process onto each subspace. The main advantage on ranking algorithms inside a subspace is that evaluation profiles are likely more correlated.
In this section, we have shown that the algorithm selection problem cannot be separated from the parameter tuning problem. This statement means that a solution to the algorithm selection issue is composed of both an algorithm and a parameter setting. We have described our twofold strategy for learning the algorithm selection based on image-content analysis and algorithm ranking. Starting from a training image set and segmentation algorithms, our approach first identifies different situations based on image-content analysis, then select the best algorithm with a mean parameter set for each identified context based on optimized parameter values. At the end of the learning process, contexts are learnt with their associated pairs $\left(A, \overline{p}^A\right)$.

3.5 Learning for semantic image segmentation

In this section, we propose an approach for semantic image segmentation based on high-level knowledge acquisition and learning. Even if the segmentation is optimized, low-level segmentation algorithms cannot reach a semantic partitioning of the image. Thus, compared to the ground truth, some regions remain over-segmented, as illustrated in Figure 8. If we can assign the right label to each region, neighbouring regions with similar labels are merged and, as a consequence, the residual over-segmentation becomes invisible. This means to be able to map region features onto a symbolic concept, i.e. a class label. We use the example-based modelling approach as an implicit representation of the low-level knowledge. This approach has been applied successfully in many applications such as detection and segmentation of objects from specific classes e.g., (Schnitman et al., 2006; Borenstein & Malik, 2006). Starting from representative patch-based samples of objects (e.g., fragments), modelling techniques (e.g., mixture of Gaussian, neural networks, naive Bayes classifiers) are implemented to obtain codebooks or class-specific detectors for the segmentation of images. Our strategy follows this implicit knowledge representation and associates it with machine learning techniques to train region classifiers. The following sub-sections describe this stage in details.



(a) original image (b) ground truth (c) segmentation with default parameters (d) segmentation with optimized parameters

Fig. 8. An example of a parameter optimization loop. The final result (d) is not perfect since some regions are over-segmented with respect to the ground truth (b).

3.5.1 Class knowledge acquisition by region annotations

In our case, region annotations represent the high-level information. This approach assumes that the user is able to gather, in a first step, a representative set of manually segmented training images, i.e. a set that illustrates the variability of object characteristics which may be found. Then, the user must define a domain class dictionary composed of k classes as $Y = \{y_1, \ldots, y_k\}$. This dictionary must be designed according to the problem objectives. For instance, y_1 = background class, y_2 = object class #1, and so on. Once Y is defined, the user is invited, in a supervised stage, to label the regions of the manually segmented images with respect to Y. From a practical point of view, an annotation is done with the help of a graphical user interface we have developed. This tool allows interacting with a region-based segmentation of an image by clicking into a region and by selecting the desired class label y (see Figure 9).

At the end of the annotation task, we obtain a list of labelled ground truth regions which belong to classes defined by the user. Since the segmentation result is not exactly the same than the manual segmentation, the next step is to map, for each training image, the labels of ground truth regions onto the regions of the region map R_I^A resulting from the segmentation of the image I with the selected algorithm A tuned with the parameter set $\overline{p}^A$, as described in section 3.4. The mapping is done by majority overlap such as for each region $r \in R_I^A$,

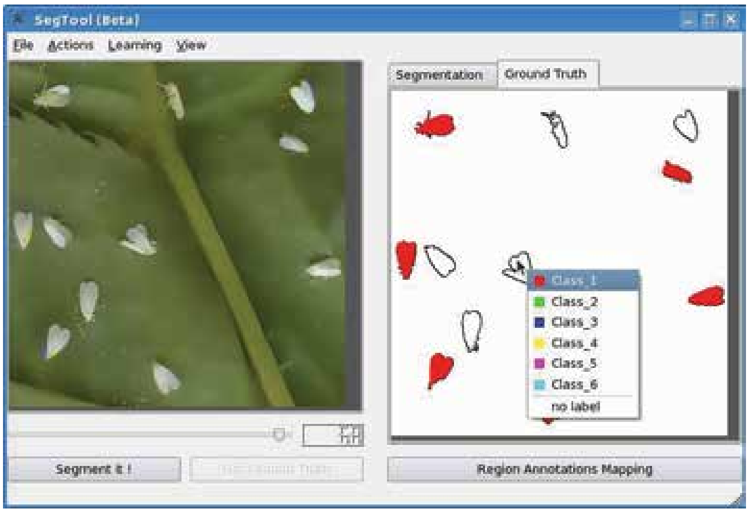


Fig. 9. Region annotations with the developed graphical tool.

$$y(r) = \begin{cases} y_i \mid i = \arg\max H_r, if \dfrac{\max h(r)}{|r|} > T \\ y_0, else \end{cases} \quad (14)$$

with $|r|$ the number of pixels of the region *r*, *T* a threshold, and $H(r) = \{h_1(r), \ldots, h_i(r), \ldots, h_k(r)\}$ the label histogram of the region *r* such that for a pixel *u* and a label y_i, $h_i(r) = card\{u \in r \mid y(u) = y_i\}, i \in 1, \ldots, k$.

If the ratio of the most represented class in the region does not reach the threshold *T* (here fixed at 0.8), the region label is set to $y_0 \notin Y$. This prevents from labelling badly segmented region as sketched in Figure 10.

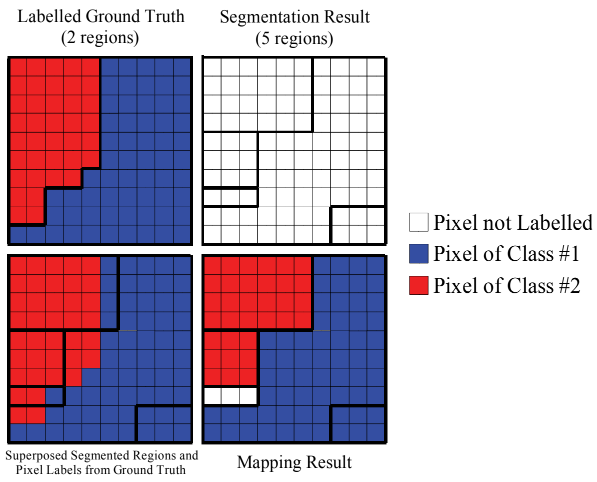


Fig. 10. Example of the mapping between a labelled ground truth regions and segmented regions.

We also denote the set of all region annotations

$$RA_{\Im} = \bigcup_{I \in \Im} \bigcup_{r \in R_I^A} \{ y(r) \mid y(r) \neq y_0 \} \tag{15}$$

and the set of all annotated regions $R_{\Im}$ such as:

$$R_{\Im} = \bigcup_{I \in \Im} \bigcup_{r \in R_I^A} \{ r \mid y(r) \neq y_0 \} \tag{16}$$

for each region, a feature vector $x(r)$ is extracted and makes with the label a pair sample noted $(x(r), y(r))$. The set of all collected pair samples from $\Im$ constitute the training data set such as:

$$T_{\Im} = \bigcup_{I \in \Im} \bigcup_{r \in R_I^A} \{ (x(r), y(r)) \mid y(r) \neq y_0 \} \tag{17}$$

$T_{\Im}$ represents the knowledge of the semantic segmentation task and is composed, at this time, of raw information. In the following section, we address the problem of knowledge modelling by statistical analysis.

3.5.2 Segmentation knowledge modelling

The first step towards learning statistical models from an image partition is to extract a feature vector from each region. But which low-level features are the most representative for a specific region labelling problem? In more general terms, which features are useful to build a good model predictor? This fundamental question, referring to the feature selection problem, is a key issue for most of the class-based segmentation approaches.

Feature Extraction

When defining a set of features for classification problems, two approaches can be considered: a first approach aims at building relevant feature sets, while a second approach more focus on the usefulness of each feature. In the first case, the choice of relevant features mostly relies on knowledge of the domain. In the second case, the goal is clearly to select features useful for building a good predictor, even if some relevant features may be excluded. We propose a trade-off approach: starting from heuristically selected features we aim at training robust region classifiers. To this end, we combine generic features, such as colour and texture and apply a feature selection algorithm.

In our approach, colour histograms represent the colour information of each segmented region. Two parameters must be set: the colour space (cs) as RGB, HSV, or XYZ, on which the histograming is applied, and the quantization parameter q which defines the number of bins. In our approach, we do not state *a priori* the relevance of one colour space against others as well as the best quantization level. We rather consider these variables as parameters of the feature selection problem.

Texture feature extraction techniques have received considerable attention during the past decades and numerous approaches and comparative studies have been presented (Reed & du Buf, 1993). The most commonly used are the grey-level co-ocurence matrices introduced by Haralick (Haralick, 1979), the Law's texture energy (Laws, 1980), and the Gabor multi-channel filtering (Jain & Farrokhnia, 1991). For the characterization of texture, we use oriented Gaussian derivatives (OGD) to generate rotation invariant feature vectors. OGD are

equivalent to the Gabor features but are computationally simpler. The basic idea is to compute the *energy* of a region as a steerable function. This energy is computed for different *power* channel, which are the result of convolving the region pixels with OGD filters of a specific order. As colour histograms, texture feature vectors depend on the parameter q.
The final feature vector representing a region is a concatenation of the feature vectors extracted from each cue. The feature extraction process is applied on each region of the annotated regions set $R_{\mathfrak{I}}$ so as to build the training data set $T_{\mathfrak{I}}$.
Following our cognitive approach of the segmentation problem, we need to avoid manually selected and tuned algorithms. At the feature selection level, this means to be able to automatically select and tune the feature extraction algorithm.

Feature Selection

The feature selection is used to reduce the number of features, remove irrelevant, redundant, or noisy data, and it brings the immediate effects of speeding up and improving the prediction performance of learning models. Since feature selection is a fertile field of research, we refer the reader to surveys (Guyon & Elisseeff, 2003; Kohavi & John, 1997; Blum & Langley, 1997) as good starting literatures. The optimality of a feature subset is measured by an evaluation criterion. Feature selection algorithms designed with different evaluation criteria broadly fall into two categories: the *filters* and the *wrappers*. Filters select subsets of features as a pre-processing step, independently of the chosen predictor. Well-known methods dedicated to this purpose are basic linear transforms of the input features like Principal Component Analysis (PCA) and Fisher Linear Discriminant Analysis (LDA). Techniques based on iterative search are also widespread as sequential forward/backward algorithms (e.g. SFFS, SBS, ReliefF). Wrappers utilize the learning machine of interest (e.g., SVM, neural networks) as a black box to score subsets of features according to their predictive power. Consequently, wrappers are remarkably universal and simple. An interesting comparative study of such feature selection algorithms can be found in (Molina et al., 2002).
The feature selection approach we propose is derived from wrappers. Our goal is to find the best feature extractor configuration which minimizes the joint classification errors of the class predictors applied on the training data set $T_{\mathfrak{I}}$. Unlike classical approaches, we act on the feature extractor parameters to generate different feature vectors, instead of reducing the feature vector itself. This approach is sketched in Figure 11. The two free parameters of our selected feature extractors are the colour space encoder for colour feature extractor, and the quantization level for both colour and texture feature extractors. The goal is to find the best combination able to induce the minimum region classification errors. The quality estimation is conducted via a cross-validation procedure which gives, for each region classifier, the classification Mean Square Error (MSE). A global MSE is then computed by averaging the MSE of each region classifiers.

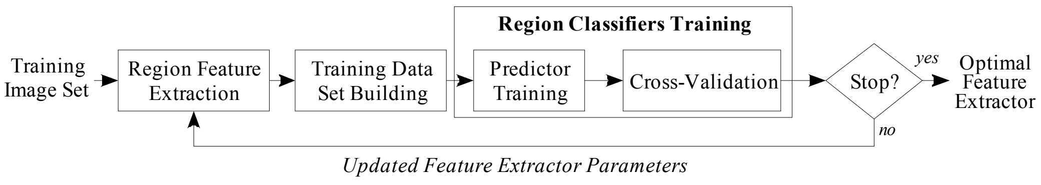

Region Classifiers Training
Training Image Set
Region Feature Extraction
Training Data Set Building
Predictor Training
Cross-Validation
Stop?
yes
no
Optimal Feature Extractor
Updated Feature Extractor Parameters

Fig. 11. Feature selection schema based on tuning of the feature extractor parameters.

We use an iterative search strategy to cover the value spaces of the two parameters *q* and *cs*. This technique guarantees to find a global optimal solution but is computationally expensive: first, it requires to run *M* x *N* x *O* region classifier training procedures, with *M* the number of quantization levels (typically equals to 256), *N* the number of color spaces, and *O* the number of classifiers to train; second, when the value of *q* increases, so does the size of the feature vector. So, to avoid an unreasonable computational time, the choice of the training algorithm must take into account this computational constraint.

Training Algorithm for Class Modelling

After extracting a feature vector for each region of the training data set, the next step is to model the knowledge in order to produce region classifiers (one classifier per class). For a feature vector *x*(*r*) and a class y_i,

$$c_i(r) = p\left(y(r) = y_i \mid x(r)\right) \tag{18}$$

with $c_i(r) \in [0,1]$ is the estimated probability associated with the hypothesis: ''feature vector x(*r*) extracted from region *r* is a representative sample of the class y_i''. The set of these trained region classifiers is noted $C = \{c_1, \ldots, c_k\}$.

A variety of techniques have been successfully employed to tackle the problem of knowledge modelling such as naives Bayes networks, decision trees or support vector machine (SVM). We propose to use SVM (Burges, 1998) as a template-based approach. SVM are known to be efficient discriminative strategies for large scale classification problems such as in image categorization (Chen & Wang, 2004) or object categorization (Huang & LeCun, 2006). SVM yields also state-of-the-art performance at very low computational cost. SVM training consists of finding an hyper-surface in the space of possible inputs (i.e. feature vectors labelled by +1 or -1). This hyper-surface will attempt to split the positive samples from the negative samples. This split will be chosen to have the largest distance from the hyper-surface to the nearest of the positive and negative samples.

We adopt a one-vs-rest multi-class scheme with probability information (Wu et al., 2004) to train one region evaluator per class. We use SVM with radial basis function as region classifiers. There are two parameters while using RBF kernels: C (penalty parameter of the error term) and γ (kernel parameter). It is not known beforehand which C and are the best for one problem; consequently some kind of model selection (parameter search) must be done. To fit the C and γ parameters, we adopt a grid-search method using 5-fold cross-validation on training data. Basically, pairs of (C, γ) are tried and the one with the best cross-validation accuracy is picked. This straightforward model selection efficiently prevents over-fitting problems. The model selection is wrapped in the feature selection schema with which it shares the cross-validation step. The training stage ends up when all combinations of ((*q*,*cs*),(C, γ)) have been tested. The one giving the lowest global classification error is picked and the region classifiers are trained a last time with this configuration.

3.6 Conclusion

In this section, we have presented our learning approach for adaptive image segmentation. We have detailed each step of the learning module for context adaptation, algorithm parameter tuning, and semantic image segmentation. The algorithm parameterisation issue is tackled with a generic optimization procedure based on three independent components. We have designed our performance evaluation metric to be broadly applicable and with a low computational cost. It allows assessing a large variety of segmentation algorithms and only relies on manual segmentations. However, further experiments need to be done to assess the performance and the accuracy of the two optimization algorithms (the Simplex algorithm and a Genetic Algorithm). The final step of the learning module is to train region classifiers to refine the segmentation according to semantic region labelling. In this task, the user must annotate the regions of the manually segmented images with class labels. Our approach is based on the discriminative power of the SVM Classifiers to ground low-level region features into symbolic classes. We have also proposed an unsupervised method for the learning of SVM and region feature extractor parameters. The goal is to optimize the performance of the classifiers without the help of the user.

4. Adaptive image segmentation

The originality of our approach is to combine bottom-up segmentation and a top-down process of region labelling in a complementary manner: in a first step, segmentation is optimized by dynamic algorithm selection and parameter tuning. Then, the bottom-up segmentation is refined thanks to region labelling to achieve the expected semantic segmentation. A new image (i.e. not belonging to the training set) segmentation is achieved by the adaptive image segmentation module in four steps (see Figure 12) using the segmentation knowledge base (learnt clusters of training images, learnt parameters, and trained region classifiers):

1. Context Prediction: a global feature vector is extracted from the image. The feature vector is classified among the previously identified clusters. The classification is obtained by assessing the distance of the feature vector to the cluster centres.
2. Algorithm selection: from the identified context, the corresponding segmentation algorithm with learnt parameters is selected.
3. Bottom-up segmentation: the image is segmented using the selected algorithm. This algorithm is tuned with the learnt parameters specific to the identified context.
4. Semantic segmentation: for each region of the segmented image, features are extracted and given as input to the region classifiers. The most probable label is assigned to the region. The final labelled partition representing the semantic segmentation of the image is returned to the user.

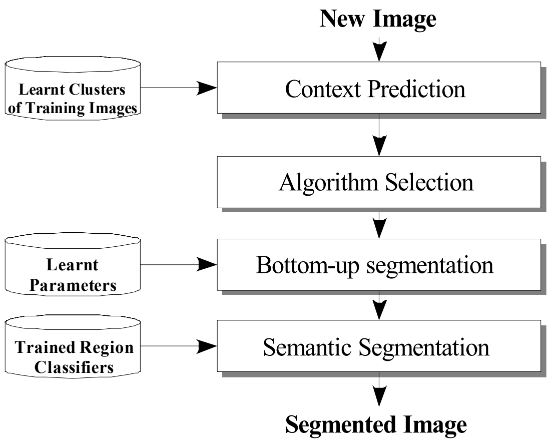


Fig. 12. Adaptive segmentation of an input image based on algorithm selection, parameter tuning, and region labelling.

For a new incoming image I not belonging to the training set, a feature vector is first extracted then classified into a cluster. The classification is based on the minimization of the distance between the feature vector and the cluster set $\{\kappa_i\}$ as follows:

$$I \in \theta_i \Leftrightarrow v(I) \in \kappa_i \mid i = \arg\min_{i\in[1,n]} dist\left(v(I), \kappa_i\right) \tag{19}$$

The pair $\left(A, \bar{p}^A\right)$ associated with the detected context θ_i is returned.
Once the algorithm is selected and tuned, the image is segmented. For each region, a feature vector is extracted using the optimized $(\hat{q}, cs)$ parameter set and given as input to each trained region classifiers c_i. Classes are scored according to the classifier responses $\{c_i(r)\}$ and finally, the returned label $y(r)$ is such as:

$$y(r) = \arg\max_{ii} c_i(r) \tag{20}$$

When all regions are labelled, neighbouring regions with the same label are merged to form a semantic partitioning of the image. This final segmentation is returned to the user.

5. Experiment for adaptive image segmentation

We present below the experimentations we have conducted to assess our framework. The first application we focus on is the segmentation of biological objects in their natural environment. More precisely, the goal is to segment white flies on rose leaves (see Figure 13). The images are acquired from a flatbed scanner. White flies are very small objects (2mm), their wings are semi-translucents, and they can be seen from different points of view. Rose leaves are highly textured with many veins and present various appearances.

5.1 Segmentation algorithms

In this section, we briefly describe the segmentation algorithms we used for our experiment. Our set is composed of algorithms reflecting different segmentation strategies as developed in section 2.1 namely region growing, split-and-merge, watershed, or thresholding techniques. The Table 1 summarizes these algorithms and gives important information concerning their free parameter with their ranges and default values provided by the algorithms' authors.

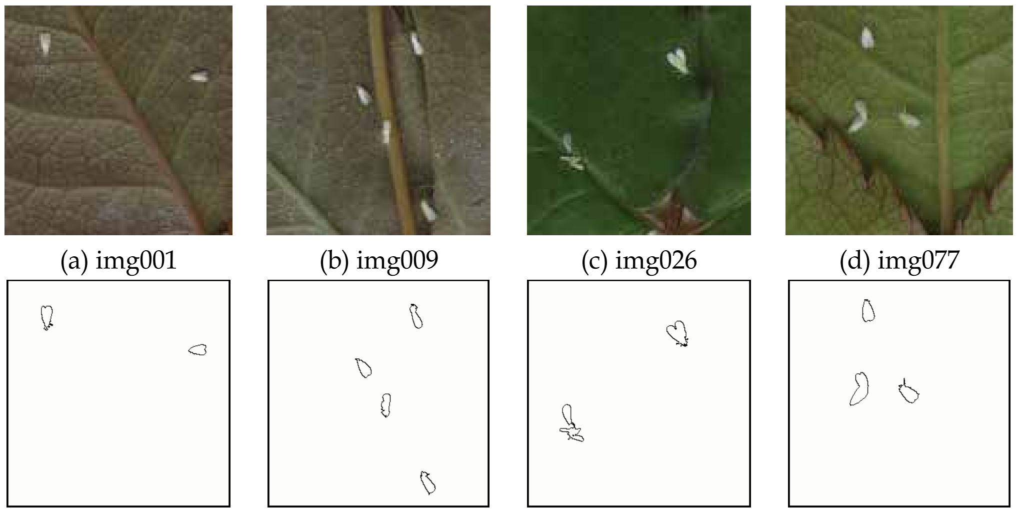
5.2 Parameter optimization assessment

Before assessing the optimization procedure, we illustrate the optimization problem with some examples of evaluation profiles. We present 1D and 2D profiles for the different segmentation algorithms (except the CWAGM which has a parameter space in R^3) for the four training images of the Figure 13. The best segmentation quality correspond to an assessment value $E_I^A = 0$. Concerning the CSC algorithm (see Figure 14), the shapes of the curves are similar for the four images and present a global minimum which falls in the same part of the parameter space. The global optima for the SRM algorithm (see Figure 15) are found in a very narrow band of the parameter space. Many local optima characterize the curves of the EGBIS algorithm (see Figure 16). The thresholding algorithm behaviour is more straightforward regarding to the obtained curves (see Figure 17). Globally, two performance levels are revealed where good performances are achieved for a large range of the parameter values. However, the global optimum is more difficult to see since the difference between the good performance level (in blue) and its level is very thin. From these observations, we can conclude that the evaluation profiles are not always convex hulls and their granularity can depend on the image. Since the Simplex algorithm does not guarantee to obtain a global optimum, we divide each parameter space into three sub-spaces and run an optimization on each sub-space. This means that 3^N optimization loops are run for a segmentation algorithm with N free-parameters. Table 2 present the optimization results of the five segmentation algorithms in terms of segmentation performance. Globally, all the algorithms reach a good level except the EGBIS algorithm, as shown in Figure 18. This result is due to the fact that this algorithm is sensitive to small gradient variations. As expected, the EGBIS has a big standard deviation (due to the presence of many local optima) whereas the thresholding one is low (due to its straightforward behaviour). We have also compared the performances of the optimization algorithms (the Simplex and the GA) with a systematic search method (third part of Table 2). By systematic, we mean an iterative search throughout the whole parameter space with a fixed sampling rate. The sampling rate depends on the dimensionality of the parameter space. The global performances of the three methods are similar with a very little advantage to the Simplex.

Algorithm	Free Parameter	Range	Default Value
CSC (Priese et al., 2002) Color Structure Code	t: region merging threshold	5.0-255.0	20.0
SRM (Nock & Nielsen, 2004) Statistical Region Merging	Q: coarse-to-fine scale control	1.0-255.0	32.0
EGBIS (Felzenszwalb & Huttenlocher, 2004) Efficient Graph-Based Image Segmentation	σ: smooth control on input image k: colour space threshold	0.0-1.0 0.0-2000.0	0.50 500.0
Hysteresis thresholding	T_{low}: low threshold T_{high}: high threshold	0.0-1.0 0.0-1.0	- -
CWAGM (Alvarado Moya, 2004) Color Watershed-Adjacency Graph Merge	m: region merging threshold n: min. region number p: min. probability for watershed threshold	0.0-200.0 1.0-100.0 0.0-1.0	100.0 10.0 0.45

Table. 1. Components of the segmentation algorithm bank, their names, and parameters to tune with range and author's default values.

To decide between the three different methods, we have compared them by considering their computational cost as described in Table 3. The systematic search is obviously the most costly method. The Simplex is the fastest method to converge apart from the CWAGM algorithm. According to the previous performance score tables, the simplex is definitively the best algorithm to optimize low dimensional parameter spaces in a few numbers of iterations. For segmentation algorithms with more than two free-parameters, the Genetic Algorithm should be preferred, requiring less iteration for the same level of performance. Note that we have limited the number of iterations — mainly for computational cost reasons—for the systematic search method to 2550 for the EGBIS algorithm and to 1250 for the CWAGM algorithm, respectively. These two algorithms are relatively slow compared to the others and the parameter space to explore is really huge, particularly for the CWAGM.


(a) img001 (b) img009 (c) img026 (d) img077

(e) gt001 (f) gt009 (g) gt026 (h) gt077

Fig. 13. Four representative training images and associated ground truth segmentations used in figure 14 to figure 17.

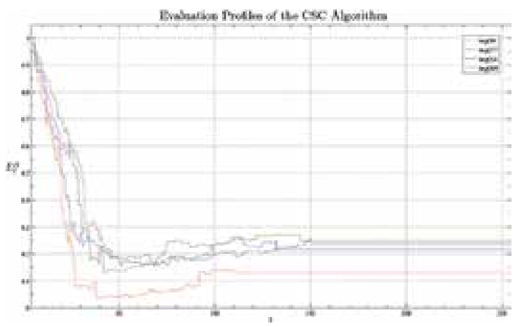


Fig. 14. Evaluation profiles of the CSC algorithm applied on the four training images presented in Figure 13.

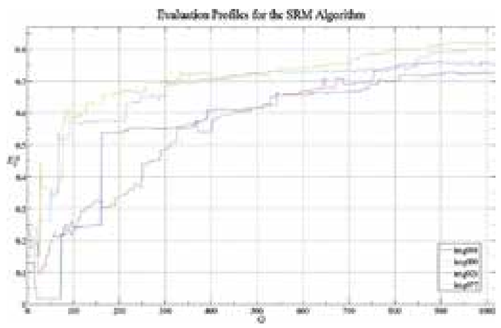


Fig. 15. Evaluation profiles of the SRM algorithm applied on the four training images presented in Figure 13.

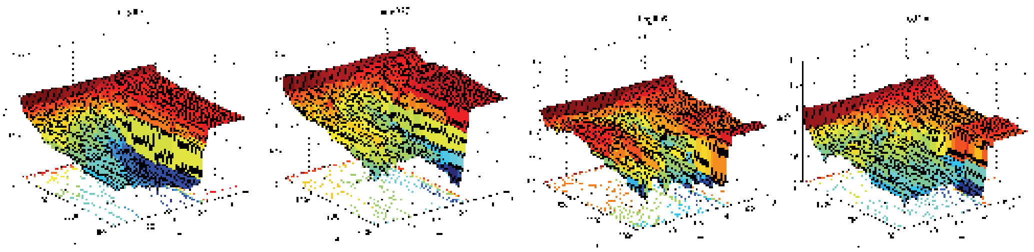


Fig. 16. Different evaluation profiles of the EGBIS algorithm applied on the four training images presented in Figure 13. t and σ are the two free parameters.

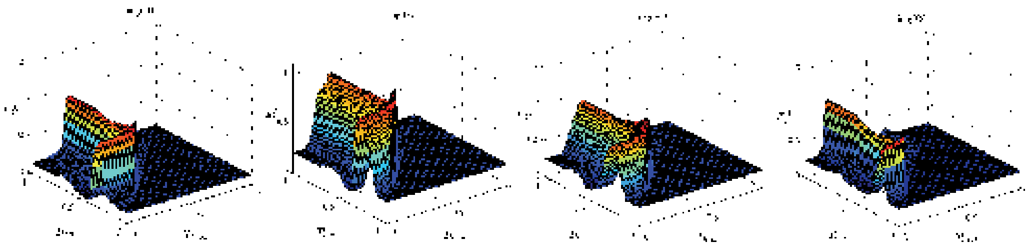


Fig. 17. Different evaluation profiles of the Hysteresis thresholding algorithm applied on the four training images presented in Figure 13. T_{low} and T_{high} are the two free parameters.

Algorithm	$E_I^A = 0$ using Simplex / GA / Iterative search			
	min	max	mean	std
CSC	0.00 / 0.00 / 0.00	0.50 / 0.46 / 0.46	0.14 / 0.13 / 0.13	0.11 / 0.10 / 0.10
SRM	0.00 / 0.00 / 0.00	0.52 / 0.48 / 0.48	0.13 / 0.12 / 0.12	0.11 / 0.10 / 0.10
THRESH	0.00 / 0.00 / 0.00	0.35 / 0.35 / 0.35	0.11 / 0.11 / 0.11	0.09 / 0.09 / 0.09
EGBIS	0.06 / 0.12 / 0.13	0.73 / 0.71 / 0.71	0.37 / 0.37 / 0.39	0.14 / 0.14 / 0.14
CWAGM	0.00 / 0.00 / 0.00	0.44 / 0.44 / 0.46	0.12 / 0.12 / 0.19	0.09 / 0.09 / 0.08

Table 2. Statistics on the optimization performances for the training image set using the Simplex algorithm, the genetic algorithm and the systematic search.

Algorithm	Mean number of iterations		
	Systematic search	Genetic algorithm	Simplex algorithm
CSC	1000	733	83
SRM	1000	734	82
THRESH	10000	840	404
EGBIS	2550	840	497
CWAGM	1250	840	1821

Table 3. Computational cost of each optimization method.

The number of iterations is also dependent of the parameterisation of the optimization algorithm. For the Simplex algorithm, it mainly depends on the *maxCalls* parameter which specifies the maximum allowed number of calls of the fitness function in an optimization loop. Figure 18 (left) shows the influence of this parameter on the convergence accuracy. We start the test on the img001 with *maxCalls* set to 3 (minimum allowed by the algorithm) and increase it up to 80. For a one-dimensional parameter space, this means that the total number of iterations will be between 9 (3×3) and 240 (3×80), for a two dimensional space between 27 ($3^2 \times 3$) and 720 ($3^2 \times 80$), and so on. The study of the graph brings us to several conclusions. The dimensionality of the parameter space to explore has to be taken into account for the setting of *maxCalls* but excessive values are useless. The study also reveals that the parameter space is not explored in the same way, depending on the segmentation algorithm. Indeed, some algorithms have parameter subspaces which induce flat evaluation profiles, as for instance the thresholding algorithm. In these sub-spaces, the Simplex converges in a few numbers of iterations. The same study is done for the GA and the results are graphically reported in Figure 18 (right). We decide to assess the GA sensitivity to the initial population size. The number of initial points is here independent of the segmentation algorithm and varies between 20 and 840. The same conclusions can be drawn. We just can add that the EGBIS algorithm brings some problem to the GA which falls in many local optima (peaks of the EGBIS curve in Figure 18 (right)).

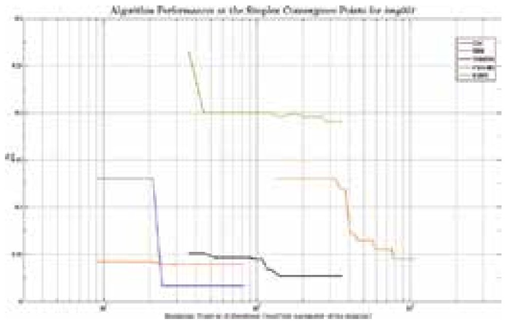
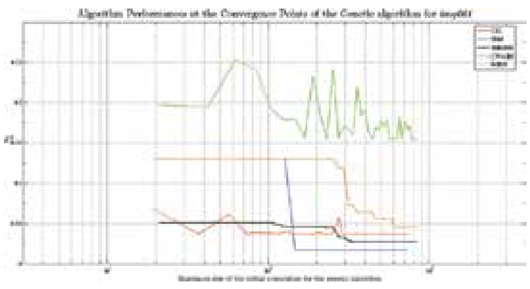



Fig. 18. Convergence accuracy of the Simplex algorithm by varying the *maxCalls* parameter and convergence accuracy of the GA by varying the initial population size.

5.3 Algorithm selection

We applied the DBScan (Ester et al., 1996) algorithm to cluster the 20 training images as described in section 3.2. We obtain two clusters of 10 images (see Figure 19 for two examples). Visually, the first cluster corresponds to the back side images of the scanned rose leaves and the second cluster to the front side images. For each cluster, mean parameter sets of the five segmentation algorithms are computed w.r.t. their performance scores. The segmentation performances of the tuned algorithms are evaluated on each training image sub set. The tuned algorithm which gets the best mean performance score for each cluster is elected. Before the last ranking step, the best algorithm for the first cluster was the Hysteresis thresholding algorithm and the best for the second cluster was the CSC algorithm. After the last ranking step, the CSC algorithm was found as the best one for the two clusters but with different parameter sets. This means that even if the thresholding algorithm performs better in individual cases, the CSC algorithm is more robust than the thresholding algorithm when tuned with a mean parameter set.

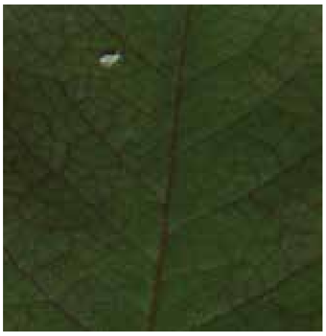
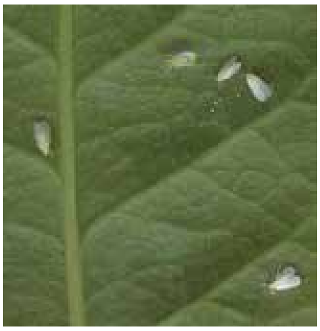



Fig. 19. Examples of images for the two identified clusters. Left = cluster 1 (front side of the leaves), right = cluster 2 (back side of the leaves).

5.4 Semantic segmentation performance assessment

For each identified image cluster, region labels of annotated manual segmentations are mapped into regions of the segmented image following the method described in section 3.5.1. Then, for each region class, a region classifier is trained with region features as input. We used our wrapper scheme detailed in section 3.5.2 to optimize the classifier performances. Three colour spaces are used in this experiment: RGB, HSV, and XYZ. The optimization of the SVM parameters increases the classifier performances of 5-10%. The best cross-validation rates are reached with q (quantization level) values superior to 50. We have also tested texture features but their performances are 10% inferior in mean than with the colour features as shown in Figure 20. Finally, the classifiers are trained a last time with the configurations giving the best cross-validation rates. The final set up of the different algorithms is then as follows (see Table 4):

Context	Seg. Algorithm	Class	Feature extractor param.		SVM param.	
	(param.)		Colour space	q	C	γ
Context 1	CSC	rose leaf	HSV	112	4	1
(light green leaves)	(41.9)	white fly	HSV	112	1	4
Context 2	CSC	rose leaf	XYZ	21	64	4
(dark green leaves)	(48.7)	white fly	XYZ	21	256	0.25

Table 4. Set up of the segmentation, the feature extractors, and the classifiers.

5.5 Final segmentation quality assessment

In this section, we present the segmentation results on the test set. We compare six different methods, comprising (parts of) our approach and a pure top-down segmentation.

- method 1: *ad hoc* segmentation, with the Hysteresis thresholding algorithm tuned with $T_{low} = 0.45$ and $T_{high} = 1.0$,
- method 2: algorithm selection and tuning based on the learnt parameters from the whole training set (CSC is the best algorithm),
- method 3: method 2 + semantic segmentation (region labelling),
- method 4: algorithm selection and tuning based on image content analysis (one algorithm with learnt parameters per context),
- method 5: method 4 + semantic segmentation,
- method 6: over-segmentation + semantic segmentation.

The over-segmentation used in method 6 is performed with the CWAGM algorithm manually tuned with a very low region merging threshold (see Figure 20).

Performance scores of the test set are summarized in Table 5. Methods 3 and 5 give the best results. This outcome is predictable since the segmentation algorithm used for the method 5 is the same (CSC) and the parameter setting for the context 1 is close to the one for the context 2. The white fly region classifier for the context 2 has been trained on few samples since there are not many white flies on the front side of rose leaves. Consequently, the classification errors for the white fly class are higher for the method 5 context 2 than for the method 3. In a biological point of view, insects prefer to live hided on the back side of the leaves, where they are better camouflaged (low contrast, not visible, etc.). Method 6 does not perform better results even if its initial over-segmentation is more precise (i.e. less missed boundary pixels) than with the CSC algorithm in methods 2 to 5.

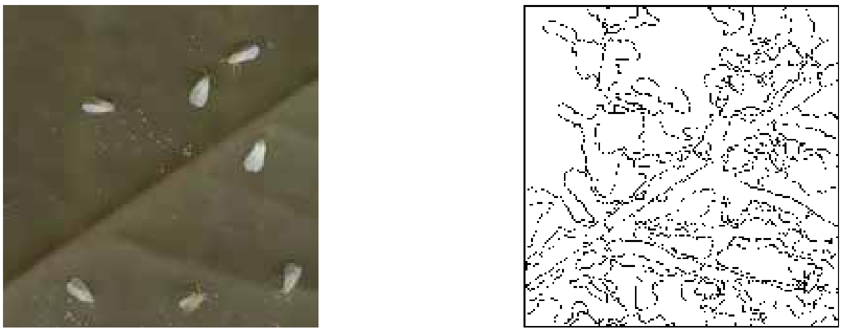


Fig. 20 Example of an initial over-segmented image used in method 6.

5.6 Evaluation on a public image database

In this section, we present evaluation results of the parameter optimization step on a public image database. The goal of the Berkeley Segmentation Dataset and Benchmark (BSDB) image database (Fowlkes & Martin, 2007) is to provide an empirical basis for research on image segmentation and boundary detection. To this end, the authors have collected 6000 hand-labelled segmentations of 300 Corel dataset colour images from 30 human subjects. The images depict natural scenes with at least one foreground object (e.g., an animal, a plant, a person, etc.). The ground truth are not labelled and the possible semantic classes are too numerous. Consequently, we do not assess the semantic segmentation part of our framework on this image database.

Method	Performance scores of segmentation of the test images			
	min	max	mean	std
1	0.00	0.35	0.09	0.08
2	0.00	0.78	0.21	0.16
3	0.00	0.65	0.12	0.14
4	0.06	0.83	0.23	0.17
5	0.06	0.62	0.12	0.14
6	0.00	0.67	0.15	0.14

Table 5. Statistics on the segmentation performances for the test set using different segmentation strategies.

The evaluation metric proposed in this image database for the benchmarking cannot be used with region-based segmentation algorithms since it relies on soft boundary maps of edge-based segmentation results (e.g. maps of gradient magnitude). We thus prefer our segmentation performance metric. For each image, several human segmentations exist (from three to eight) with different levels of refinements. We have decided to select the finest ones. Then, for each segmentation algorithm of our algorithm bank and for each image, algorithm parameters are optimized thanks to the selected manual segmentation. As previously done in section 5.2, we have compared the optimized segmentation achieved with the three optimization algorithm based on: the Simplex algorithm, the Genetic Algorithm, and a systematic search (see Table 6). Globally the three optimization algorithms perform in mean comparable results. This confirms the reliability of our parameter tuning approach for this image database.

Algorithm	$E_I^A = 0$ using Simplex / GA / Iterative search			
	min	max	mean	std
CSC	0.29 / 0.37 / 0.25	0.50 / 0.46 / 0.46	0.14 / 0.13 / 0.13	0.11 / 0.10 / 0.10
SRM	0.25 / 0.23 / 0.23	0.52 / 0.48 / 0.48	0.13 / 0.12 / 0.12	0.11 / 0.10 / 0.10
THRESH	0.19 / 0.19 / 0.38	0.35 / 0.35 / 0.35	0.11 / 0.11 / 0.11	0.09 / 0.09 / 0.09
EGBIS	0.21 / 0.20 / 0.34	0.73 / 0.71 / 0.71	0.37 / 0.37 / 0.39	0.14 / 0.14 / 0.14
CWAGM	0.22 / 0.22 / 0.50	0.44 / 0.44 / 0.46	0.12 / 0.12 / 0.19	0.09 / 0.09 / 0.08

Table 6. Statistics on the optimization performances for the training image set using the Simplex algorithm, the genetic algorithm and the systematic search.

7. Conclusion and future work

In this chapter, we address the problem of image segmentation with a cognitive vision approach. More precisely, we study three major issues of the segmentation task in vision systems: context adaptation, selection of an algorithm and tuning of its free parameters, according to the image content and to the application needs. Most of the time, this tedious and time-consuming task is achieved by an expert in image processing using a manual trial-and-error process. Recently, some attempts at automating the extraction of optimal parameters of segmentation have been made but they are still too application-dependent. The re-usability of such methods is still an open problem. We have chosen to handle this issue with a cognitive vision approach. Cognitive vision is a recent research field which proposes to enrich computer vision systems with cognitive capabilities, e.g., to reason from *a priori* knowledge, to learn from perceptual information, or to adapt its strategy to different problems.

We propose a supervised learning-based methodology for off-line configuration and on-line adaptation of the segmentation task in vision systems. The off-line configuration stage requires minimal knowledge to learn the optimal selection and tuning of segmentation algorithms. In an on-line stage, the learnt segmentation knowledge is used to perform an adaptive segmentation of images. This cognitive vision approach to image segmentation is thus a contribution for the research in cognitive vision. Indeed, it enables robustness, adaptation, and re-usability faculties to be fulfilled.

Finally, by addressing the problem of adaptive image segmentation, we have also addressed underlying problems, such as feature extraction and selection, and segmentation evaluation and mapping between low-level and high-level knowledge. Each of these well-known challenging problems is not easily tractable and still demands to be intensively considered. We have designed our approach (and our software) to be modular and upgradeable so as to take advantage of new progresses in these topics.

The brittleness of our approach to unknown situations is currently its major drawback. This concerns the context analysis level as well as the segmentation level. The concerned algorithms are the DBScan algorithm for image-content clustering and the SVMs for the semantic segmentation. Currently, neither the clustering algorithm nor the SVMs are able to adapt dynamically to new training data: the learning process must be run again on the whole training data set. The use of incremental machine learning techniques should be useful to fulfil the property of continuous learning. The main idea of incremental learning for unforeseen situations is to dynamically adapt the clustering/classification method w.r.t. to the classification error of new input data. In our problem, unexpected situations can be identified thanks to the estimates of the context probability and the estimates of the SVM classification probabilities. The use of an adaptive classification algorithm using robust incremental clustering as proposed in (Prehn & Sommer, 2006) will then allow a dynamic update of the cluster and create new ones if necessary.

8. References

Abdul-Karim, M.-A.; Roysam, B.; Dowell-Mesfin, N. M.; Jeromin, A.; Yuksel, M. & Kalyanaraman, S. (2005). Automatic selection of parameters for vessel/neurite segmentation algorithms, *Transactions on Image Processings*, Vol. 14(9), pp. 1338–1350

Alvarado Moya, J. P. (2004). Segmentation of color images for interactive 3d object retrieval, *Ph.D. thesis*, Technical University of Aachen

Bahnu, B.; Lee, S. & Das, S. (1995). Adaptive image segmentation using genetic and hybrid search methods, *Trans. on Aerospace and Electronic Systems*, Vol. 31(4), pp. 1268–1291

Blum, A. L. & Langley, P. (1997). Selection of relevant features and examples in machine learning, *Artif. Intell.*, Vol. 97(1-2), pp. 245–271

Borenstein, E. & Malik, J. (2006). Shape guided object segmentation, in: *Proc. of the Int. Conf. on Computer Vision and Pattern Recognition*, Vol. 1, pp. 969–976

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery,* Vol. 2(2), pp. 121–167

Chen, Y. & Wang, J. Z. (2004). Image categorization by learning and reasoning with regions, *Journal of Machine Learning Research*, Vol. 5, pp. 913–939

Cinque, L.; Corzani, F.; Levialdi, S.; Cucchiara, R. & Pignalberi, G. (2002). Improvement in range segmentation parameters tuning, in: *Proc. of the Int. Conf. on Pattern Recognition*, Vol. 1, pp. 10176

ECVISION (2005). A research roadmap of cognitive vision, *Technical report*, Project IST-2001-35454

Ester, M.; Kriegel, H.-P.; Sander, J. & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise, in: *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining*, Portland, pp. 226–231

Everingham, M.; Muller, H. & Thomas, B. (2002). Evaluating image segmentation algorithms using the pareto front, in: *Proc. of the Eur. Conf. Computer Vision*, Vol. 2353(4), pp. 34–38

Felzenszwalb, P. F. & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation, *Int. Journal on Computer Vision*, Vol. 59(2), pp. 167-181

Fowlkes, C. & Martin, D. (2007). The berkeley segmentation dataset and benchmark, http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Gelasca, E.; Salvador, E. & Ebrahimi, T. (2003). Intuitive strategy for parameter setting in video segmentation, in: *Visual Communications and Image Processing*, Vol. 5150, pp. 998–1008

Goldberg, D. E. (1989). Genetic Algorithms in Search, *Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Boston, MA, USA

Guyon, I. & Elisseeff, A. (2003). An introduction to variable and feature selection, *J. Mach. Learn. Res.*, Vol. 3, pp. 1157–1182

Haralick, R. (1979). Statistical and structural approaches to texture, in: *Proc. of the IEEE*, Vol. 67

Huang, F. J. & LeCun, Y. (2006). Large-scale learning with svm and convolutional nets for generic object categorization, in: *Proc. of the Int. Conf. on Computer Vision and Pattern Recognition*, pp. 284–291

Jain, A. K. & Farrokhnia, F. (1991). Unsupervised texture segmentation using gabor filters, *Pattern Recognition*, Vol. 24(12), pp. 1167–1186

Kohavi, R. & John, G. H. (1997). Wrappers for feature subset selection, *Artificial Intelligence*, Vol. 97(1-2), pp. 273–324

Laws, K. (1980). Textured image segmentation, *Ph.D. thesis*, Univ. Southern California

Lucchese, L. & Mitra, S. (2001). Color image segmentation : A state-of-the-art survey, Vol. 67, *Indian National Science Academy,* New Dehli, pp. 207–221

Mao, S. & Kanungo, T. (2000). Automatic training of page segmentation algorithms : An optimizatin approach, in: *Proc. of the Int. Conf. on Pattern Recognition*, Barcelona, Spain, pp. 531–534

Maurer, C. R. J.; Qi, R. & Raghavan, V. (2003). A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions, *Trans. on Pattern Anlasys and Machine Intelligence.*, Vol. 25(2), pp. 265–270

Mezaris, V.; Kompatsiaris, I. & Strintzis, M. (2003). Still image objective segmentation evaluation using ground truth, in:*Proc. of the Workshop on Information and Knowledge Management for Integrated Media Communication*, Prague, Czech Republic, pp. 9–14

Molina, L. C.; Belanche, L. & Nebot, A. (2002). Feature selection algorithms : A survey and experimental evaluation, in: *Proc. of the Int. Conf. on Data Mining,* Japan, pp. 306–313

Nelder, J. & Mead, R. (1965). A simplex method for function minimization, *Computer Journal*, Vol. 15, pp. 1162–1173

Nock, R. & Nielsen, F. (2004). Statistical region merging, *Pattern Analysis and Machine Intelligence*, Vol. 26(11), pp. 1452–1458

Pal, N. R. & Pal, S. K. (1993). A review on image segmentation techniques, *Pattern Recognition*, Vol. 26(9), pp. 1277–1294

Peng, J. & Bahnu, B. (1998). Delayed reinforcement learning for adaptive image segmentation and feature extraction, *Systems, Man, and Cybernetics*, Vol. 28(3), pp. 482-488

Pignalberi, G.; Cucchiara, R.; Cinque, L. & Levialdi, S. (2003). Tuning range image segmentation by genetic algorithm, *EURASIP Journal on Applied Signal Processing*, Vol. (8), pp. 780–790

Prehn, H. & Sommer, G. (2006). An adaptive classification algorithm using robust incremental clustering, in: *Proc. of the Int. Conf. on Pattern Recognition*, pp. 896–899

Priese, V.; Rehrmann, L. & Sturm, P. (2002). Color structure code. URL http://www.uni-koblenz.de/

Reed, T. R. & du Buf, J. M. H. (1993), A review of recent texture segmentation and feature extraction techniques, *CVGIP: Image Underst.*, Vol. 57(3), pp. 359–372

Rosenberger, C.; Chabrier, S.; Laurent, H. & Emile, B. (2005). Unsupervised and Supervised Segmentation evaluation, in book: *Advances in Image and Video Segmentation*, Yu-Jin Zhang editor, Tsinghua University, Beijing, China, chapter XVIII

Schnitman, Y.; Caspi, Y.; Cohen-Or, D. & Lischinski, D. (2006). Inducing semantic segmentation from an example, in: *ACCV*, Vol. 3852, Springer-Verlag, pp. 393–384

Skarbek, W. & Koschan, A. (1994). Colour image sementation - a survey, *Technical report*, Technical University of Berlin

Wu, T.; Lin, C. & Weng, R. (2004). Probability estimates for multi-class classification by pairwise coupling, *The Journal of Machine Learning Research*, Vol. 5, pp. 975–1005

Xia, Y.; Feng, D.; Rongchun, Z. & Petrou, M. (2005). Learning-based algorithm selection for image segmentation, *Pattern Recognition Letters*, Vol. 26(8), pp. 1059–1068

Yasnoff, W.; Mui, J. & Bacus, J. (1977). Error measures for scene segmentation, *Pattern Recognition*, Vol. 9, pp. 217–231

Zhang, Y. (1996). A survey on evaluation methods for image segmentation, *Pattern Recognition*, Vol. 29(8), pp. 1335–1346

Zhang, Y. & Luo, H. (2000). Optimal selection of segmentation algorithms based on performance evaluation, *Optical Engineering*, Vol. 39, pp. 1450–1456

Zhang, H.; Fritts, J. E. & Goldman, S. A. (2008). Image segmentation evaluation: A survey of unsupervised methods, *Computer Vision and Image Understanding*, Vol. 110(2), pp. 260-280

An Introduction to the Problem of Mapping in Dynamic Environments

Nikos C. Mitsou and Costas S. Tzafestas
National Technical University of Athens, School of Electrical and Computer Engineering, Division of Signals Control and Robotics,Zografou Athens, Greece

1. Introduction

Robotic mapping comprises one of the most important problems in the field of robotics. During the past two decades, a large number of algorithms have been proposed in order to solve the problem of constructing valid models of the robot environment. As a result, highly accurate maps of large-scale indoor and outdoor environments have been constructed thus far. There are still, though, much to be done in order to achieve fully autonomous mobile robots capable of mapping any kind of environment (structured or unstructured, static or dynamic).

In this chapter, we discuss the problem of mapping dynamic environments, an issue that remains open and is extremely active nowadays. *Dynamic environments* are real world environments where moving objects (e.g. humans, robots, chairs and doors) change their positions over time. Widespread mapping algorithms developed in the past are based on the assumption that the environment remains static during the robot exploration phase. Thus, these algorithms provide imprecise results when applied in non-stationary environments. The need to map these environments has led in the development of new algorithms that are designed to exploit the dynamics of the environments towards efficient mapping. These algorithms have given so far promising results.

Through this chapter, we examine the problem of dynamic environments through the mapping point of view. Two issues that are strongly connected to mapping are (a) localization, the process of estimating the position and the orientation of the robot and (b) navigation, the generation of valid paths for the robot. They are both of great importance and remain open fields of research especially when applied on dynamic environments. However, in this chapter we concentrate on the mapping problem and refer to the other two problems only when necessary. Our effort is towards providing the ideas behind the algorithms discussed in this chapter and avoid the mathematical details and formulas. We urge the interested readers to consult the referenced papers in order to gain a better insight on the techniques discussed in this chapter.

The outline of the chapter is as follows: In Section 2, we present the mapping problem for static environments, so as to make the reader familiar with the concepts of the mapping problem. Next, in Section 3 we move to the problem of mapping in dynamic environments. More specifically, we discuss the main difficulties of the problem and present a number of methodologies that are common cases for dealing with it. In Section 4, a number of solutions are presented. We explain how artificial intelligence ideas are applied in some state of the art dynamic environment mapping algorithms. We discuss some recently published algorithms that apply statistical methodologies to identify the static and different aspects of the dynamic areas of the environment. Finally, in Section 6, we discuss the open issues and challenges of mapping in dynamic environments.

2. The problem of mapping in static environments

Robotic mapping is the problem of creating a valid model of the robot's environment. The robot explores the unknown environment, collects a number of sensor measurements and creates a spatial representation of its world. This representation might contain three different types of areas:

a. the *static areas*, which are the areas that remained occupied during the robot exploration process (e.g. walls, heavy furniture),
b. the *unoccupied areas,* which are the areas that do not contain any object (e.g. passages) and
c. the *unexplored areas,* which are the areas that we obtain no information about their occupancy.

There exist a number of different sensor types that are used during the mapping procedure and can be divided into two categories: the sensors that are used to identify the internal state of the robot and those that provide information about the robot's environment. In the first category, common examples are encoders, accelerometers, gyroscopes and GPS receivers (for outdoor navigation only) that provide information about the robot's motion and position. Ultrasonic sensors, laser range finders, bumpers, acoustic sensors and cameras are some examples of the second category which are used to detect and identify objects that lay in their field of view.

The main difficulty in mapping lies in the fact that none of the above sensors can be precisely accurate. Also, the reliability of the sensors might depend on external parameters that change over time (e.g. motor encoders accuracy depends on the slippery of the ground which is different on grass and on marble and the ultrasonic measurements will be different when the detected objects are made of glass and of wood). Thus, there will always be errors in the sensor measurements. Unfortunately, small uncertainties in some measurements can yield in huge errors in the estimation of the robot position which will lead in inaccurate maps.

In order to deal with these uncertainties, most algorithms use probabilities to describe both the robot position and the robot's environment. A number of different probabilistic approaches for the preservation of the uncertainties on the robot position and a number of world-modelling techniques can be found in the literature (more on localization and mapping can be found in (Thrun, 2002)). Some common mapping techniques are the occupancy grid (examples in (Gutmann & Konolige, 1999), (Birk & Carpin, 2006)), the geometric maps (examples in (Latecki & Lakaemper, 2006), (Zhang & Ghosh, 2000)) and the landmark representation (examples in (Larionova et al., 2006), (Suau, 2005)). They all have given successful results under specific assumptions on static environments.

Next, we describe in detail the occupancy grid technique, since it is the most common and widely used technique for modelling the robot environment. Moreover, the occupancy grid technique is the most important among the few proposed techniques that can be applied in both static and dynamic environments (there also exist a few works where the static areas of the dynamic environment are modelled using landmarks like in (Wang et al., 2007), (Andrade-Cetto & Sanfeliu, 2002) or using line segments like in (Angelov et al., 2004)).

2.1 The occupancy grid technique for modelling static environments

The occupancy grid algorithm (Moravec & Elfes, 1985) (Moravec, 1988), introduced in 1985, is the first robotic mapping technique. The basic idea of the occupancy grid is to split the environment into a finite number of cells. This way, the difficult problem of estimating the model of the environment is decomposed into a number of easier problems, those of estimating the occupancy state of every cell. So, instead of searching in the space of all possible maps (a difficult task as there exist an arbitrary number of different maps), we search in the space of the occupancy probability for the estimation of the state of every cell (a fairly easier task).
An example of an occupancy grid is depicted in Fig. 1 below, where the color denotes the probability of occupancy (black cells are occupied, white cells are free whereas in case of the grey areas we do not possess any knowledge about their occupancy).

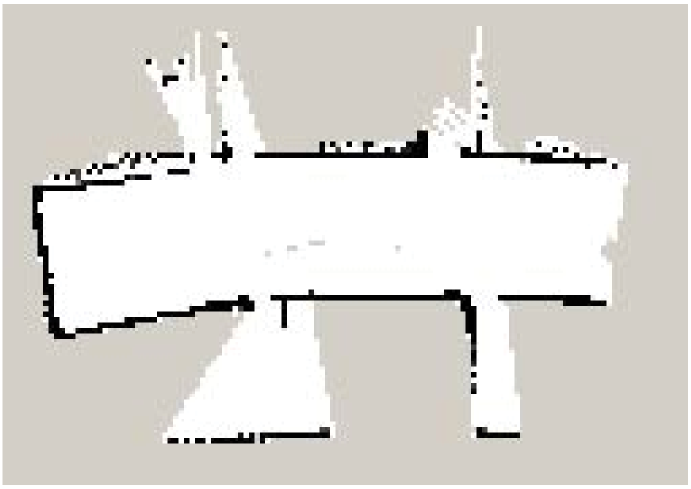


Fig. 1. Example of an occupancy grid map

In mathematical point of view, the occupancy grid algorithm can be formulated as finding the map m of the environment that maximizes the probability:

$$p(m|u_{1:t}, z_{1:t}) \tag{1}$$

where u refers to the control motion commands of the robot and z comprises the sensor measurements received during the interval [1, t].
By following the assumption that the occupancy of a cell is independent of the occupancy of its neighboring cells, the previous probability can be transformed into the product of the probabilities of the occupancy of every cell:

$$p(m|u_{1:t}, z_{1:t}) = \prod p(m_i \mid u_{1:t}, z_{1:t}) \tag{2}$$

where m_i is the i-th cell of the grid.

The assumption of the independency of the values of neighboring cells, though wrong, can be used in order to simplify the required calculations (in (Thrun, 2003), however, the authors overcome this assumption by using forward sensor models).
In order to update the occupancy probability of a cell, we can use the log-likelihood sensor model as in (Arbuckle et al, 2002):

$$\mathrm{R}_{\mathrm{i,t}} = \log \frac{p(z_t \mid s_i = occ)}{p(z_t \mid s_i = emp)} \tag{3}$$

where $p(z_t \mid s_i = state)$ denotes the probability of the observation z_t given that the state of the cell i equals to *state*. So, the update of the occupancy value R_i of the i-th cell of the grid can be calculated as in the following formula:

$$\mathrm{R}_{\mathrm{i}} = \sum \mathrm{R}_{\mathrm{i,t}} \tag{4}$$

In practice, it has been shown that by using the occupancy grid modeling of the environment, high quality maps of static environments can be generated.

3. The problem of mapping in dynamic environments

In the previous section, we presented a short introduction to the problem of mapping in static environments. However, as already stated, real world environments are dynamic rather than static. For example, people might be walking, robots might be moving, doors might be opened or closed and chairs might be repositioned. Tackling the mapping problem in dynamic environments is a more difficult and challenging task. We do not only have to determine the position of the robot and the map of the environment, but also to identify possible dynamic objects in the environment. Additionally, we should take advantage of this knowledge in order to enhance our perception of the environment and create a better model of the world.
Applying traditional algorithms from the static environments domain like the occupancy grid technique presented above will not create a precise representation of the environment. The reason is that most mapping algorithms continuously update their grids in order to adapt to the current state of the environment, so the dynamic areas will most probably be misclassified either as occupied or as free areas. The place where a chair existed, for example, might be identified as a static area even though the chair had its position changed during the mapping phase and it might be repositioned in a currently unoccupied place in the future. Even worse, there might be cases where the last observed part of an object might be assumed as static while the rest of the object might be assumed as free. A failure example for the occupancy grid technique is shown in Fig.2 below where two doors are partially identified. Their state changed while the robot was partially observing them, so some cells of the grid adapted to the new state while the others remained in the previous state of the door.
So far, we have seen that dynamic areas can lead in the creation of spurious objects in the under construction model of the environment. The existence of dynamic areas will also cause problems in the localization procedure, since matching these areas with the current sensor measurements during a scan match phase will give indeterminable results. The localisation procedure will most probably fail if, for example, the robot tries to match a range scan with the area of a door observed earlier to be closed while now is open.

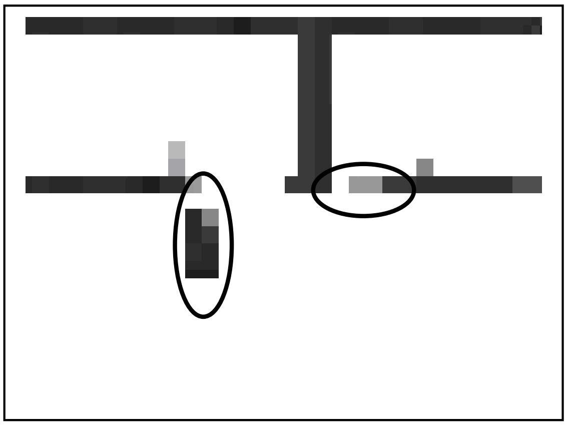


Fig. 2. Failure on mapping two doors. The doors (pointed in circles) are partially drawn.

These problems occur due to the fact that the occupancy grid algorithm does not have the ability to remember the past states of the environment. Every cell stores only one occupancy value which converges to the current observed state of the cell. If this state changes, the occupancy value will adapt to the new state, "forgetting" any past value. Moreover, the false assumption that the cells evolve independently to each other should not be used in this case as the assumption of the independency on the evolution of the cells' occupancies can lead to the problem of a partially modelled object discussed earlier (c.f. Fig. 2.).

In order to deal with these problems, several approaches have been proposed which are based on the occupancy grid structure and either make use of this structure as is or extend it and make use of the modified occupancy grid structure. By using these techniques, one can store more information about the robot's environment (including information about the past as well), which can then be processed in order to extract additional knowledge about the dynamics of the environment.

There exist three basic categories of approaches:

i. *Occupancy grids with different timescales*, which uses a number of occupancy grids where each grid is updated at a different rate.

ii. *Temporal occupancy grid* mapping, which extends the occupancy grid to efficiently preserve the history of the evolution across the time axis.

iii. The *static-dynamic grids* mapping approach which uses two grids to differentiate the static from the dynamic areas of the map.

Below we present each category in detail.

3.1 Occupancy grids with different timescales

In this approach, we use more than one occupancy grids (Arbuckle et al, 2002) (Biber & Duckett, 2005). Every occupancy grid has a particular timescale, which indicates how often it is refreshed (e.g. every 20 seconds). We can store all the grids generated so far in a list or just preserve the grids that are updated in the current step of the algorithm. By examining the occupancy values of the cells at the different occupancy grids (meaning at different timescales) we can conclude on the nature of the cell. For example, we can identify that a cell is static if it is occupied in all timescales or that it is free if it is not occupied at all the timescales. We can also conclude that a cell was occupied by a moving object if it was occupied only at one occupancy grid while on the others it remained unoccupied.

One approach that makes use of this technique is the so-called temporal occupancy grid (TOG) (Arbuckle et al, 2002). It was introduced as a way to classify cells based on their time properties of occupancy. The TOG is an extension to the common occupancy grid through the time dimension. It can be modeled as a matrix with two spatial dimensions, one time dimension and a number of additional dimensions equal to the number of different timescales being considered. So, instead of preserving only one probability value for every cell of the grid, a number of different values are preserved, where each value represents the occupancy of the cell at a specific time period and at a specific timescale. The update of the occupancy value at time t and at timescale Δt is performed using the following formula:

$$\mathrm{R}_{\mathrm{i,t,\Delta t}} = \sum_{t-\Delta t<t'\leq t} R_{i,t'} \tag{5}$$

By examining the occupancy value of a cell at the different timescales we can identify the nature of the cell. If it is occupied at every occupancy grid, then it refers to a *static object*. If it is not occupied in any of the occupancy grids, then it refers to a *free area* of the environment. If it is found to be occupied at some of the occupancy grids, then it was occupied by a *moving object*. It can be modeled as a occupancy grids under specific timescales, we can also extract the path of that moving object.
The main problem of this approach lies in the fact that the optimal number of different timescales, an important parameter of the algorithm, cannot be computed in a formal way but must be intuitively selected by the user.

3.2 Extended temporal occupancy grid

With the structure explained earlier, we do not make efficient use of the available memory, as we preserve more occupancy values than needed. For a particular occupied cell, we store the occupied probability for that cell as many times as the number of the occupancy grids in the TOG structure. In order to make a more efficient use of the occupancy values, an extended temporal occupancy grid (eTOG) has been proposed in (Mitsou & Tzafestas, 2007). Although they share the same name, the two structures are different in their nature. Instead of preserving more than one occupancy grids as in TOG, in eTOG we use only one grid. Every cell of the grid, however, contains an index structure, the so-called time index (Elmasri et al., 1990) that keeps track of the occupancy probability of the specific cell. In this way the complete history of the occupancy probability of every cell is stored in this structure. By using a grid of n x n time indexes to form a forest of time indexes, we can preserve the complete history of the changes of the environment.
Time index is a special case of B+ tree index (a widely used index structure in the database domain) (Ooi & Tan, 2002) that is used for storage and retrieval of values that are valid during specific time periods. The time index was specifically designed for indexing of temporal data. The time dimension is represented using the concept of time intervals. A time interval $[t_i, t_j]$ is a set of consecutive (equidistant or not) time points, where t_i is the first point and t_j is the last point of the interval. A single time point t can be represented as [t, t], i.e. both start and end points are the same. An example of a time index structure appears in Fig.3.
The time index differentiates from the B+ tree index in the fact that due to the monotonic nature of time, deletions never occur while updates occur in an append mode. So, new entries will always be inserted in the rightmost node of the tree and the complexity of the

insertion will always be O(1). When the rightmost node is full, a new node is created and the changes propagate upwards, just as in a B+ tree index structure (an extensive comparison of available time indexes can be found in (Salzberg & Tsotras, 1999)).

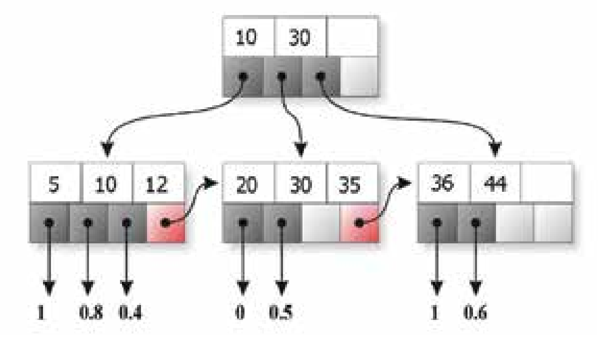


Fig. 3. A time index structure storing probabilities of occupancy

We have to keep in mind that instead of time indexes, we could use simple linked lists if we are not interested in efficiently traversing the occupancy history and we want to avoid the rebalancing cost of the tree.

When a new sensor measurement arrives, the affected cells are extracted and for every cell, the new occupancy value is calculated and inserted into the underlying time index of the cell. The insertion of a new value *v* (assuming that the current time point is t_c) follows specific rules:

- If the time interval is the rightmost node finishes at time point t_j and t_c-t_j> *thres* then a new node is added to the tree, that stores a time interval with starting and ending points at t_c and with value *v*.
- Else if the value *v* equals to the value stored on the rightmost node then update the end point of the time interval to t_c.
- Else if the value *v* is not equal to the value stored on the rightmost node then a new node is added to the tree which points to a time interval with starting and ending points at t_c and with value *v*.

In this way, we can preserve the complete history of the evolution of the occupancy of the cells of the environment. In order to identify the nature of a cell, we traverse the lead nodes of the underlying time index. If the stored probabilities are all equal to the occupied or to the free state, then the cell belongs to a *static object* or to a *free area* respectively. On the other hand, if the probabilities are mixed then the cell has been occupied at some time periods by a *dynamic object*.

The main drawback of this approach is the fact that although in static environment, the memory needed is almost equal to the memory needed by the common occupancy grid, in dynamic environments, the required memory might be extremely large. Indeed, the memory increases with the size of the dynamic effects.

3.3 The static-dynamic grids approach

The most common approach in mapping dynamic environments is the preservation of (at least) two occupancy grids (Tanaka & Kondo, 2006), (Wolf & Sukhatme, 2004), (Wang et al, 2003). In the first grid, only the static areas of the environment are stored while in the second grid, the dynamic objects are preserved. In the first grid, the occupancy probability of a cell

represents the probability of a static entity being present at that cell. In the same manner, a cell in the second grid has an occupancy probability that indicates the probability of a moving object's existence in that cell.
Such a structure is employed in (Wolf & Sukhatme, 2004). In order to specify which areas are static and which dynamic, the authors use a simple differentiation technique. The static parts of the environment never change their position so they can be used as a reference to determine which sensor readings are generated by static and dynamic objects. Two recursive formulas based on the Bayes rule for updating the probabilities of the cells of the two grids are presented. Also, two fuzzified inverse sensor models are introduced. The first models the probability of a cell being static given the current observation and the previous probability of occupancy and the second the probability of a cell being dynamic given the current observation and the previous probability of occupancy. No historical information is preserved about the occupancy of the dynamic map. This means that only the current position of the dynamic objects can be available. In (Wolf & Sukhatme, 2005), a third grid was also maintained, that contained the positions of landmarks and was used during the localization process.

4. Algorithmic implementation of mapping techniques for dynamic environments

In order to develop successful maps for dynamic environments, we should first understand the notion of a dynamic environment map and how it differentiates from the traditional map produced in stationary environments. We categorize the objects in the robot environment into the following three groups:

- *Static objects*, these are the objects that do not change their position over time (e.g. walls, beds or locked doors)
- *Low dynamic objects*, these are the objects that appear in a specific number of places (e.g. chairs or doors).
- *High dynamic objects*, these are the objects that move arbitrarily in the environment and can be found in many different positions (e.g. humans).

Ideally, to successfully map a dynamic environment, we should create different maps for every category of the objects presented above (or one map with three different layers). Firstly, we must create a map of the static objects of the robot environment. This map could be an occupancy grid map of the environment where every cell that is detected to remain occupied during the robot exploration phase is marked as static. There should also be a second map for storing the low dynamic objects of the environment. This map should contain the current state of these objects and also their other possible configurations as they are detected during the sensor acquisition phase (e.g. the chair is positioned in area A and can be found in the areas B and C). Finally, a third map can be created that will show the current state of all the high-dynamic objects of the environment. In this map, we could also mark the path of the moving objects as observed and identified by the robot. The combination of these three maps will create the current state of the environment and will also contain all the knowledge extracted by the sensor measurements.
During the last few years, three lines of research have emerged towards mapping in dynamic environments:

- algorithms that aim in the preservation of *an up-to-date map of the environment* using aging techniques

- algorithms that focus on the creation of the possible different configurations of the environment
- algorithms that map populated environments

The algorithms proposed so far are partial solutions to the problem as they deal with the problem of mapping focusing only on some of the aspects described above. In the following subsections we are going to discuss the most common approaches for every line of research.

4.1 Mapping using aging techniques

In this category, several approaches have been proposed that utilize aging techniques in order to create valid maps of the current state of the environment. Previous sensor measurements are slowly forgotten in order to preserve at any time point a valid map of the current state of the robot's environment. The goal of these algorithms is to create a number of up-to-date models of the world, rather than to differentiate between static and dynamic areas. These methods are adaptive, by means that they adapt the map of the environment so as always to reflect the current understanding of the robot about the environment.

The main difficulty of this category lies in the fact that the under construction map model must be able to adapt to changes of the environment. When a door closes, for example, the map should change the state of the particular cells that refer to the closed door configuration in a fairly short period of time. However, these algorithms should also be able to detect possible sensor errors or the existence of a high dynamic object in the environment (e.g. a human that passes by the door). This means that a false sensor measurement or a measurement of a human passing by a door should not change the state of the corresponding cells. The algorithm must be able to identify the error and ignore the measurement. So, in other words, the generated map must contain the static and the low-dynamic areas of the world but ignore the high dynamic objects.

In (Zimmer, 1995), a system that can dynamically learn and update the topology of a map is presented. The system preserves a model that is able to adapt to changes of the environment. In (Yamauchi and Beer, 1996), an adaptive place network is used to model the robot's environment. The network is able to change the confidence for every link and links with low confidence can be removed from the map. In (Andrade-Cetto & Senafeliu, 2002), landmarks are used to model the map of the environment whose positions are preserved with the use of Kalman Filters. Landmarks can disappear from the map if they are removed from the environment.

The most representative approach in this category of mapping algorithms can be found in (Biber & Duckett, 2005). The basic representation of a map m_{ti} at time point t_i is considered to consist of a set $S(t_i)$ of n measurements. The new map $S(t_{i+1})$ at time point t_{i+1} is calculated by an update rule that depends on the update rate u of the map: Remove $u*n$ randomly chosen measurements from $S(t_{i+1})$ and replace them with $u*n$ randomly chosen measurements received during the time interval $[t_i, t_{i+1}]$.

The environment is split into sub-maps and a number of different timescale Occupancy grid maps for every sub-map are used. Short-term maps, meaning maps that are updated frequently, react quickly to changes and only a few sensor measurements are required to forget an old estimation. On the other hand, long-term maps, meaning maps that have a low update ratio, are updated less frequently and do not react to temporary changes. They only adapt to consistent changes of the environment. These map models have increased accuracy

towards the static parts of the environment as sensor errors are in a way ignored since a large number of samples are required to update an already learned feature of the world.
This algorithm belongs to the same category with the TOG algorithm described earlier. It uses multiple grids of different timescales (although the number of Occupancy grids preserved by TOG is greater) with different however update schema from the TOG update schema. Also, it encounters the same problem with the TOG algorithm, which is the selection of the number of the different timescales that are needed to successfully map the robot's environment. Unfortunately, there is no clear answer to that problem. The more maps we preserve, the better we model the environment as we obtain more models of it (short-term and long-term maps). However, with the increasing number of maps, more memory is needed to store those maps.

4.2 Detecting possible environment configurations

In this section, we will present three algorithms that are used to identify all possible configurations of the environment. These approaches are motivated by the fact that many dynamic objects appear only in a limited number of different positions. Consider for example, the doors in an office environment, which are usually either open or closed. The knowledge of the possible positions of a low dynamic object can be used to enhance the localization process. If the robot identifies a door to be closed then it will expect that this door will not be found also open. A formalistic framework for localization and environment configuration selection can be found in (Stachniss & Burgard, 2005).
The three algorithms presented bellow follow the common occupancy grid assumption, that the occupancy of a cell evolves independently of the occupancy of its neighbouring cells but try to overcome this assumption by post processing the sensor measurements and searching for associations among cells. The first algorithm, *ROMA* uses an Expectation Maximization technique (Dempster et al., 1977) to identify different occurrences of the same objects in the robot environment. The second algorithm, (Stachniss & Burgard, 2005) uses a fuzzy clustering algorithm to identify common areas in a number of different occupancy grid maps. The third algorithm, (Mitsou and Tzafestas, 2007) uses an extended Temporal Occupancy grid and exploits the temporal behaviour of the cells to create group of cells that behave in the same manner. An extended survey on clustering can be found in (Jain et al., 1999).
The first two algorithms use the shape similarity to detect low dynamic objects while the third one makes use of the temporal similarity. The last two algorithms use clustering to extract knowledge from the environment. The former one uses the collected occupancy grids as instances of the clustering algorithm while the last one uses the occupancy evolutions of the cells for the same reason.
Apart from these three algorithms, other algorithms also exist that deal with the same problem. For example, in (Schulz & Burgard, 2001), a probabilistic algorithm is presented to identify the state of dynamic objects in the environment. In (Avots et al, 2002), the authors use particle filters and conditional binary Bayes filters to estimate the state of doors in the environment. In both these works, the environment is assumed to be predefined. Also, in (Anguelov et al, 2004), an Expectation Maximization algorithm is used to detect and model doors. Additionally to the laser range device used in the previous algorithms, images from a camera mounted on the robot are used. Every object in the world has a specific shape and color. The advantage of this algorithm is that an object can be identified to be a door even if it did not move during the experiment (assuming that all doors bare the same color).

4.2.1 The ROMA algorithm

The *Robot Object Mapping Algorithm* (ROMA), also found as *Dynamic Occupancy Grid Mapping Algorithm* (DOGMA), (Biswas et al., 2002) aims at the identification of moving objects and the extraction of their models, at learning in other words, the models of the low dynamic objects of the environment. To learn these models, we assume that the robot maps the environment at different points in time, between which the configuration of the environment may have changed. Each map is represented as a static occupancy grid map.

By using a simple map differencing technique, we can detect the dynamic areas of the environment. The result is a list of "snapshots" of low dynamic objects, each represented by a local occupancy grid map. Two such snapshots of the same low dynamic object (e.g. a chair) might be completely different from each other. The object might be translated and rotated in a different way in the two snapshots. In order to group the snapshots that correspond to the same object, a modified Expectation Maximization (EM) algorithm is applied.

The algorithm uses the following two steps:

a. Step 0: The robot observes the dynamic environment and generates snapshots of the low dynamic objects.
b. Step 1: The EM algorithm is applied:
    a) E step: Create correspondences between the different snapshots based on the estimated models of the objects.
    b) M step: Create the models of the objects based on the correspondences.

The analysis of these steps follows below.

Step 0. Create initial snapshots

The robot explores the environment (which is assumed to remain static or to change slowly) and acquires sensor values. Based on these measurements, the areas of the environment that contain dynamic objects are identified with the use of a simple differencing technique. The areas that are occupied in some maps and free in other indicate low dynamic objects. From these areas, snapshots of these objects are extracted.

Step 1. Apply the EM algorithm

The EM algorithm is applied in order to associate snapshots on different occupancy maps with specific objects.

In the E step of the EM algorithm, correspondences between different snapshots at different points in time are established. Based on the models found on the M step, we associate snapshots at different time points and we estimate the best rotational and translational parameters so that the snapshot will match with the estimated model of the moving object.

In the M step, these probabilistic correspondences are used to regenerate new estimates for the object models. We combine the snapshots that correspond to the same object and we create the most probable model of the object.

By iterating between the E and the M step, we will finally converge to the correct models of the low dynamic objects and their correspondences in the occupancy grid maps. However, as we do not know in advance the number of moving objects in the robot environment, we have to run EM in a number of times with different number of possible moving objects, starting from the lower bound of the number of objects (the maximum number of objects identified in a single map). The number of objects that maximizes the probability of the detected snapshots given the models of the moving objects while minimizes a penalty factor (large numbers of objects are assumed to be less possible) is selected as the best choice. An advantage of the *ROMA* is the fact that it can identify objects that can take any arbitrary position in the environment (assuming that they move slowly). It does not require the objects to be positioned in specific areas, since it is based on the shape similarity of the moving objects. However, the fact that it identifies an object's occurrences based on its shape can lead to incomplete results. Objects that are partially observed (if for example placed next to walls or to other objects) might not be successfully correlated to their other occurrences in the environment.

The Hierarchical Object Mapping algorithm

An extension of the *ROMA* algorithm was presented in (Anguelov et al., 2002). The basic purpose of this work were to extract not only the model of the moving objects but also possible object templates. This approach is able to generalize across different object templates, as long as they model objects of the same type. It uses again an extension of the EM algorithm. It has been shown that this approach performs better than *ROMA*.

4.2.2 Stachniss & Burgard approach

In this approach (Stachniss & Burgard, 2005), the information about changes in the environment during robot exploration was used to estimate possible spatial configurations (examples in Fig. 4). A number of snapshots of the environment at different time intervals were collected and clustering was used to create groups of common snapshots.

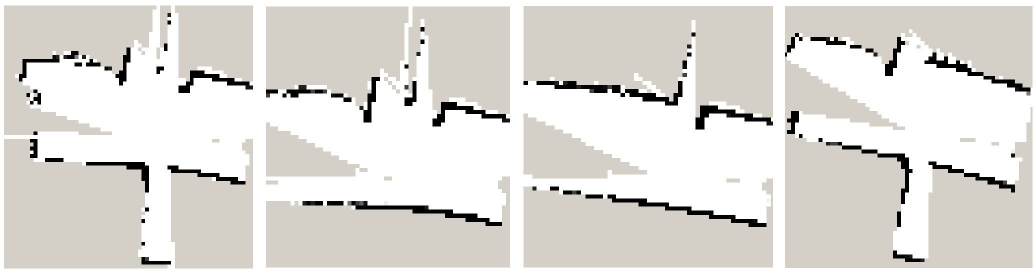


Fig. 4. Different configurations of an environment with three doors.

With this approach, low dynamic objects that lay in the robot environment can be identified. This algorithm assumes that the environment remains static during every time interval of exploration.

The algorithm uses the following two steps:

a. Step 0: The robot observes the dynamic environment at different time intervals and creates a number of sub-areas.
b. Step 1: For each sub-area, we apply a clustering technique to create similar configurations into groups.

The analysis of these steps follows below.

Step 0. Data Acquisition – Sub-areas creation

The robot explores the environment (which is assumed to remain static or to change slowly) and acquires sensor values. Based on these measurements, the areas of the environment that contain dynamic objects are identified. The environment is then segmented into local areas, called sub-maps, so that each sub-map contains a small number of dynamic areas.

Step 1. Clustering occupancy grids

For every sub-map created in the previous step, the occupancy grids are generated from the sensor values collected earlier. Each occupancy grid captures the state of the sub-map at a given time period. By clustering the occupancy grids for every sub-map separately, we can create groups of similar occupancy grids. Each such group denotes a different configuration of the sub-map.

It is a necessity to create sub-maps of the environment. If we did not, we would have to store a number of maps that would be exponential in the number of dynamic objects. This means that we would need a huge number of occupancy maps of the whole environment to successfully cluster them into groups. Instead, by splitting the environment into sub-areas, we need a smaller number of smaller occupancy grid maps.

In order to cluster the occupancy grids, we transform the grids into a vector of probability values from 0 to 1 with the additional value of ξ. The ξ value represents an unobserved cell. When comparing two such vectors *a* and *b*, the following similarity measure has been shown to give good results:

$$\mathrm{d}(\alpha,b) = \sum_i \begin{cases} (a_i - b_i), a_i \neq \xi \wedge b_i \neq \xi \\ 0, a_i = \xi \wedge b_i = \xi \\ e, otherwise \end{cases} \tag{6}$$

where *e* is a number close to zero.

Then, a fuzzy k-means algorithm is used (Duda et al, 2001). In order to estimate the correct number of clusters (not known in advance), we iterate over the number of clusters and compute in each step a model using the fuzzy k-means algorithm. In each iteration, we create a new cluster initialized using the input vector which has the lowest likelihood under the current model. We then evaluate every model with the Bayesian Information Criterion (BIC) (Schwarz, 1978). The model with the biggest BIC is selected as the best representation of the environment.

4.2.3 Extended temporal occupancy grid algorithm

The key idea of this approach is to use an extended Temporal Occupancy Grid to preserve the occupancy history of the environment. The values stored in the grid are processed to extract information about the possible configurations of the environment.

With this approach, low dynamic objects can be detected without any prior knowledge of the object shape or motion. This algorithm does not assume that the environment remains static during the robot exploration. On the contrary, in order to correctly detect the dynamic objects, it demands that these objects move fairly enough in order to be distinguished from the static or the high dynamic objects.

The algorithm uses three steps:

a. Step 0: The robot explores the dynamic environment and fills the extended Temporal Occupancy grid with occupancy values.
b. Step 1: Find possible object configurations by grouping neighbouring cells that follow the same pattern of occupancy evolution
c. Step 2: Identify which configurations found in the previous step belong to the same object.

The analysis of these steps follows.

Step 0. Data Acquisition
The robot explores the environment and acquires sensor values. These values are stored in the eTOG. The environment is assumed to be active during the exploration, meaning that low dynamic objects must change their positions thus different configurations of the environment are observed. One single pass of the environment is sufficient to identify all the low dynamic objects that were observed to move.
Step 1. Find possible configurations
In order to identify different configurations of the environment, we safely assume that a low dynamic object covers more than one cell in the environment (a door, for example, might cover three or more cells). Such an object falls into different configurations/ states (in the previous example, the states could be open and closed). In order to detect these configurations, we search through the history of all cells to find neighbouring cells that change with the same motif. These cells correspond to one of the possible configurations of the object. For example, in the case of a door, all cells of the closed door configuration would have the same values, regardless of the door's state (occupied if the door is closed, free if it is opened). To find cells that change with the same motif, we treat the values in the leaf nodes of the cells as Time Series (collections of observations made sequentially in time) that describe the cells occupancy. A single time series describes the evolution of the occupancy of the corresponding cell over time. Similar time series indicate similar cell occupancies. Thus, in order to find a single configuration of a moving object, we aim in finding neighbouring cells with similar time series. To do so, a clustering algorithm is applied.
To cluster time series, we need to define an appropriate distance function. There exist various similarity measures in the data mining community. A simple yet effective function is the Minkowski distance:

$$\mathrm{D}_{\mathrm{Mink}}(T_1,T_2)=\sqrt[p]{\sum_{i=1}^{n}|t_{1,i}-t_{2,i}|^p} \tag{7}$$

Any other distance measure can be applicable.
At first, an agglomerative hierarchical clustering algorithm (Jain et al., 1999) is applied. In hierarchical clustering the data is not partitioned into a particular cluster in a single step. Instead, a series of partitions takes place. In each partition, two clusters are combined according to their distance to create a new cluster. The number of clusters is not given as an input. The algorithm keeps merging clusters until a given threshold (in the distance of two just merged clusters) is reached. The choice of an appropriate threshold is intuitive and depends on the data to be clustered.
The agglomerative clustering will generate a clustering where every cluster contains similar time series. Most probably each cluster will contain only one moving object. However, in the extreme case that two dynamic objects have similar change rate, the cluster will contain time series of more than one dynamic object. To avoid this case, we post process the generated clusters in order to create sub-clusters of both similar and neighbouring time series. To do so, we apply a second clustering algorithm, a DBSCAN algorithm (Ester et al., 1996) in order to group neighbouring time series. DBSCAN is a density based clustering that creates groups of data objects. Each of the objects in a group has at least one other object with distance less than a given threshold. DBSCAN does not require the number of clusters as an input to the algorithm and can create groups of arbitrary shapes. So, the DBSCAN algorithm

gets as input the cell coordinates of the time series of every agglomerative cluster extracted by the first clustering and groups them into dense clusters. Eventually, after the two clustering algorithms, the low-dynamic object configurations will have been found.

Step 2. Associate Configurations

In this step, we search for correlations among the configurations found in the previous step, for patterns that represent different configurations of the same object. The simpler case is the case of two-state objects, i.e. objects that can appear in two different positions. In such objects, their configurations are complementary; when the one is occupied the other one is free. Thus, in order for two different patterns to belong to the same object, their combined Time Series must contain the occupied value at any time.

Following the same rationale, we can search for objects with three, four, five etc different configurations. The difference is that we have to combine and evaluate more than two Time Series. In order for the algorithm to correctly associate the different patterns, the moving objects must change their positions with different time rates. If, for example, two doors follow the same motion pattern, the algorithm will be confused in the pattern association step. We have to keep in mind that the quality of the associations drops with the increase of the number of the states of the objects.

Experimental results can be found in Fig.5.

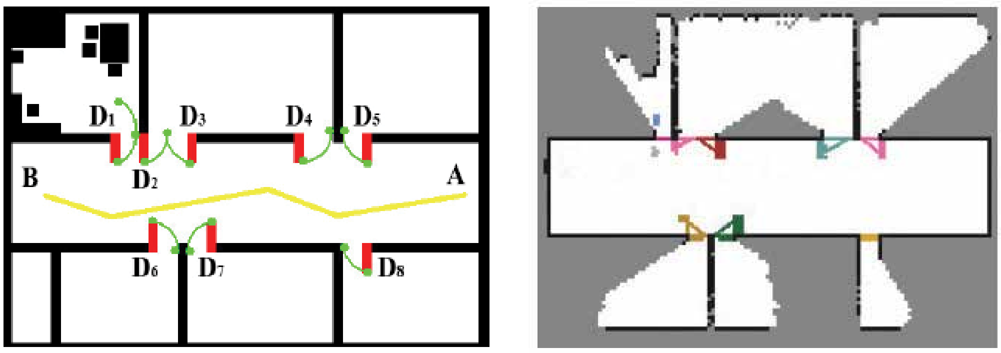


Fig. 5. The environment (yellow line indicates the robot path) and the experimental results of the eTOG algorithm. Associations between different configurations are indicated with a line that connects them.

4.3 Mapping in populated environments

Populated environments are highly dynamic environments where a lot of people move within them, e.g. crowded museums, metro stations at rush hour, supermarkets. The main difficulty in mapping those environments is the fact that many sensor measurements correspond to dynamic objects rather than static areas of the environment. So, the extraction of the model of the environment might contain spurious objects that reduce the quality of the generated maps.

There exist two categories of methods for dealing with populated environments both aiming at discovering the high dynamic objects and omitting them during the mapping phase. Their difference lies in the fact that the algorithms in the first category apply filters in order to track the moving objects while those in the second category apply statistical methods to identify the measurements that correspond to dynamic objects. There exist a variety of tracking techniques in the literature. In this section, we present a short introduction on techniques that have been applied successfully in the problem of mapping populated environments.

Regarding the tracking category, a number of filtering techniques have been used. When a single object is being tracked, its state can be estimated using algorithms such as the Kalman Filter, Extended Kalman Filter and Particles Filter (Bellotto & Hu, 2007).

Multiple objects tracking can be seen as the same problem with higher complexity. With multiple objects there is the additional difficulty of distinguishing one target from the other (data association). The complexity of solving the data association problem grows exponentially with the number of targets. Moreover, the number of different objects in the environment is also unknown and has to be derived from the observations. Once the number of objects is known, the data association allows the evolution of the filter associated to each object.

A common approach consists of using one Kalman Filter per target and solves data association independently for each track. This method is simple but as it examines each moving object separately it can lead to multiple associations of the single measurement with different moving objects. To deal with this problem, several approaches have been proposed, including Joint Probabilistic Data Association Filters (JPDAFs) (Hähnel et al., 2003a), Multi Hypothesis Tracking (MHT) (Mucientes & Burgard, 2006) and Particle Filtering. Also, combinations of these methods have been proposed such as the SJPDAFs that combine particle filters with JPDAF. Further information can be found in (Hähnel et al., 2003a), (Hähnel, 2004).

A framework for the solution of the detection and tracking of moving objects (*DTMO*) problem is presented in (Wang & Thorpe, 2002). The authors use the static-dynamic grids approach (c.f. Section 3.3) in order to detect the moving objects in the robot's environment. In order to track the objects, they apply a matching technique between the current state of the environment and the previous one stored in the grid. When the best match between the two states is found, information about the moving object can be extracted. In their next work (Wang et al., 2003), they present a Bayesian formulation of the Simultaneous Localization and Mapping problem (*SLAM*) with the *DTMO* problem. In order to track moving objects, a motion based detector is used. Then a Multiple Hypothesis Tracking method is applied to find the associations between current and previously found dynamic objects. When the associations are found, we can conclude on the motion of this moving object and predict its future position.

Regarding the second category of solutions, filtering techniques are used in (Fox et al., 98) to detect humans in the vicinity of the robot (on a known environment). A distance filter is applied to filter out all measurements that are found to be shorter than expected. Also, an entropy filter is implemented to measure the uncertainty of the measurements. Those measurements that are found to increase the uncertainty of the system are ignored.

In (Hähnel et al., 2003c), an EM algorithm was used to learn which measurements correspond to static objects. In the expectation step, an estimation is made on which measurements might correspond to static parts of the environment while in the maximization step, the position of the robot is calculated with respect to the considered static areas of the estimation step. By iterating between the two steps we can generate high quality maps with significantly less dynamic objects that if we used a common algorithm for static environments. This technique can be also applied in dynamic environments with the same results.

5. Conclusions and outlook

In this chapter, we have presented, in short, the major algorithms in the field of mapping dynamic environments. We categorize the objects in the robot environments into three categories (static, low dynamic and high dynamic objects) and for each category, we present a number of solutions proposed so far in the literature. Special attention is paid on the occupancy grid structure and its variations that have been applied on dynamic environments giving promising results. Of course, it was not possible to cover all the available techniques or examine relative to mapping issues, such as localization and navigation.

Summarizing, we can state that there exist a number of promising algorithms for mapping in dynamic environments. These algorithms are able to create valid maps of the static areas of the world and detect and model (low or high) dynamic objects in the robot environment. Nevertheless, a large number of challenging issues still remains to be solved. First of all, many algorithms that are presented in this work are not *real-time*. They make use of iterative techniques (e.g. EM, clustering) with execution times that depend on the size of the data. The necessity for real-time algorithms is obvious. A robot must be able to identify its environment rapidly in order to react on time. This need is more urgent while navigating outdoors. *Outdoor mapping* is an issue that remains open. Streets, parks and garages are some examples of places where a mobile robot can be of extreme usefulness. However, these environments pose new challenging issues. Their rate of change, complexity and size make many existing algorithms inapplicable and impose the need for new algorithms suitable for the special characteristics of such environments (some examples can be found in (Nuchter et al., 2007) and (Yoon et al., 2007)).

In order to deal with these problems, an interesting direction that has to be investigated is the combination of computer vision with laser data. So far, only a few algorithms have applied techniques borrowed from the computer vision field e.g. (Anguelov et al., 2004), (Yoon et al., 2007). Vision, however, can provide a wealth of information about the state of the environment (such as humans walking by the robot or chairs in the middle of a room) and it can be really helpful in the task of object identification. The combination of vision and laser data can provide important information about the robot environment.

Another way to gain more information about the objects that surround the robot is to acquire 3D laser range data of them e.g. (Hahnel et al., 2003b), (Ryde & Hu, 2006), (Harati & Siegwart, 2007). Humans possess a complete prior knowledge of the model of the objects in the environment and can easily distinguish two different objects even when they are partially observed in contrast to modern robots. To bridge the gap between human perception and robot perception of the environment, we can utilize 3D laser data that provide more information including information about the objects' shapes.

To conclude, the problem of mapping in dynamic environments is a really challenging and extremely active research field in robotics. In the future, we will be able to design robots that will be capable of moving among humans in their own environments without interrupting human activities. In order to do that, a milestone that has to be reached is a complete solution to the problem of mapping dynamic environment.

6. Acknowledgments

This work was supported by the European Commission and the Hellenic General Secretariat for Research and Technology, under Measure 3.3 of the Operational Programme

"Information Society" in the 3rd Community Support Framework (Project name: DIANOEMA, ID:35).

7. References

Andrade-Cetto, J. & Sanfeliu, A., (2002), Concurrent Map Building and Localization on Indoor Dynamic Environments, International Journal on Pattern Recognition and Artificial Intelligence, Vol 16, pages 361-374

Anguelov, D.; Biswas, R.; Koller, D. Limketkai, B. & Sebastian Thrun (2002), Learning Hierarchical Object Maps Of Non-Stationary Environments With Mobile Robots, Eighteenth Conference on Uncertainty in Artificial Intelligence (pp. 10-17)

Anguelov, D.; Koller, D.; Parker, E. & Thrun, S. (2004), Detecting and modeling doors with mobile robots, In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)

Arbuckle, D.; Howard, A. & Mataric. M. J. (2002). Temporal occupancy grids: a method for classifying spatio-temporal properties of the environment, IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 409–414, Lausanne, Switzerland

Avots, D.; Lim, E.; Thibaux, R. & Thrun. S.(2002) A probabilistic technique for simultaneous localization and door state estimation with mobile robots in dynamic environments, Proceedings of the Conference on Intelligent Robots and Systems (IROS), Lausanne, Switzerland

Bellotto, N. & Hu, H. (2007), People Tracking with a Mobile Robot: a Comparison of Kalman and Particle Filters, 13th IASTED International Conference on Robotics and Applications (RA 2007), Germany

Biber, P. and Duckett, T. (2005) Dynamic maps for long-term operation of mobile service robots, In Robotics Science and Systems

Birk, A., Carpin, S., (2006) Merging Occupancy Grid Maps From Multiple Robots, PIEEE(94), No. 7, pp. 1384-1397

Biswas, R.; Limketkai, B.; Sanner, S. & Thrun, S. (2002) Towards Object Mapping in Dynamic Environments With Mobile Robots, Conference on Intelligent Robots and Systems (IROS), Lausanne, Switzerland

Dempster, A.P.; Laird, N.M. & Rubin. D.B., (1977), Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, Series B, pages 1–38

Duda, R.; Hart, P.; & Stork, D. , (2001), Pattern Classification. Wiley-Interscience

Elmasri, R.; Wuu, G. T. & Kim. Y. J. (1990). The time index: An access structure for temporal data, 16th VLDB, pages 1–12

Ester, M.; Kriegel, H.-P.; Sander, J. & X. Xu, (1996), A density-based algorithm for discovering clusters in large spatial databases with noise. In Second International Conference on Knowledge Discovery and Data Mining, pages 226– 231, Portland, Oregon

Fox, D.; Burgard, W.; Thrun, S. & Cremers, A.B, (1998), Position estimation for mobile robots in dynamic environments, Proceedings of the AAAI Fifteenth National Conference on Artificial Intelligence

Gutmann, J. & Konolige, K., (1999), Incremental mapping of large cyclic environments, the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), pages 318–325, Monterey, California

Hähnel, D. (2004), Mapping with Mobile Robots. PhD thesis, Fakultät für Angewandte Wissenschaften, Universität Freiburg

Hähnel, D.; Schulz, D. & Burgard, W., (2003a) Mobile robot mapping in populate environments, Advanced Robotics, Volume 17, pages 579-597(19)

Hähnel, D.; Thrun, S. & Burgard, W. (2003b), An Extension of the {ICP} Algorithm for Modeling Nonrigid Objects with Mobile Robots, Sixteenth International Joint Conference on Artificial Intelligence (IJCAI), Mexico

Hähnel, D., Triebel, R., Burgard, W. & Thrun, S., (2003c), Map Building with Mobile Robots in Dynamic Environments, In Proceedings of the International Conference on Robotics and Automation (ICRA), pages 1557-1563

Harati, A. and Siegwart, R. (2007), Orthogonal 3D-SLAM for Indoor Environments Using Right Angle Corners, The 3rd European Conference on Mobile Robotics (ECMR) 2007, Freiburg, Germany

Jain, A. K.; Murty, M. N. & P. J. Flynn, (1999), Data clustering: a review, ACM Computer Surveys, pages 264–323

Larionova, S.; Marques, L. & de Almeida, T. , (2006) Detection of Natural Landmarks for Mapping by a Demining Robot, IEEE International Conference on Intelligent Robots and Systems (IROS), pages 4959-4964

Latecki, L. J. & Lakaemper, R. (2006), Polygonal Approximation of Laser Range Data Based on Perceptual Grouping and EM, IEEE International Conference on Robotics and Automation (ICRA), Orlando, Florida

Mitsou, N. & Tzafestas, C. (2007), Temporal Occupancy Grid for mobile robot dynamic environment mapping, in the 15th IEEE Mediterranean Conference on Control and Automation, MED'07, Athens, Greece

Moravec, H. P. (1988). Sensor Fusion in Certainty Grids for Mobile Robots, AI Magazine, Vol. 9, No. 2, pp. 61-74

Moravec, H. P. & Elfes, A. E. (1985), High Resolution Maps from Wide angle Sonar, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'85), pp. 116-121, St. Louis, Missouri

Mucientes M. & Burgard W., (2006), Multiple Hypothesis Tracking of Clusters of People, 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China

Nuchter, A.; Lingemann, K.; Hertzberg J. & Surmann, H., (2007), 6D SLAM—3D mapping outdoor environments, Journal of Field Robotics

Ooi, B. C. & Tan. K. L. (2002). B-trees: Bearing fruits of all kinds, Thirteenth Australasian Database Conference (ADC2002), Melbourne, Australia

Ryde, J. & Hu, H. (2006), Mutual localisation and 3D mapping of cooperative mobile robots, The 9th International Conference on Intelligent Autonomous Systems (IAS-9), Tokyo, Japan, pages 217-224

Salzberg, B. & Tsotras, V. (1999), Comparison of access methods for time-evolving data, ACM Computing Surveys (CSUR)

Schulz, D. & Burgard, W., (2001), Probabilistic state estimation of dynamic objects with a moving mobile robot, Robotics and Autonomous Systems, pages 107–115

Schwarz, G., (1978), Estimating the dimension of a model, The Annals of Statistics 6(2)

Stachniss, C. & Burgard, W. (2005). Mobile robot mapping and localization in non-static environments, Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, USA

Suau, P., (2005), Robust Artificial Landmark Recognition Using Polar Histograms, Intelligent Robotics (IROBOT 2005) Chapter 7 - pages 455-461

Tanaka, K. & Kondo, E., (2006) Towards Real-Time Global Localization in Dynamic Unstructured Environments, Special Issue on Advanced Technology of Vibration and Sound pp.905-911

Thrun, S. (2002), Robotic Mapping: A Survey, Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann

Thrun, S. (2003), Learning Occupancy Grid Maps with Forward Sensor Models, Journal of Autonomous Robots, vol. 15, pages 111-127

Wang, C.-C. & Thorpe. C, (2002), Simultaneous Localization and Mapping with Detection and Tracking of Moving Objects. In IEEE International Conference on Robotics and Automation (ICRA'02)

Wang, C.-C.; Thorpe. C, & Thrun. S, (2003), Online Simultaneous Localization and Mapping with Detection and Tracking of Moving Objects: Theory and Results from a Ground Vehicle in Crowded Urban Areas. In IEEE International Conference on Robotics and Automation (ICRA'03)

Wang, H.; Hou, Z & Tan, M., (2007), Mapping Dynamic Environment Using Gaussian Mixture Model, 6th IEEE International Conference on Cognitive Informatics (ICCI'07)

Wolf, D. & Sukhatme, G. (2004), Online simultaneous localization and mapping in dynamic environments, IEEE International Conference on Robotics and Automation, pages 1301–1306

Wolf, D. & Sukhatme, G. (2005), Mobile Robot Simultaneous Localization and Mapping in Dynamic Environments, Journal of Autonomous Robotics, Volume 19, pages = 53-65

Yamauchi, B. & Beer, R. (1996), Spatial learning for navigation in dynamic environments. IEEE Transactions on Systems, Man and Cybernetics, Special Issue of Learning Autonomous Robots, pages 496– 505

Yoon, S.; Sung-Kee, P.; Choi, H. D.; Kim, S. & Kwak, Y. K. (2007), ViSion-Based Outdoor Simultaneous Localization and Map Building Using Compressed Extended Kalman Filter, European Control Conference (ECC07), Kos, Greece

Zhang, L. & Ghosh, B.K., (2000), Line segment based map building and localization using 2D laserrangefinder, IEEE International Conference on Robotics and Automation, ICRA '00, pages 2538-2543

Zimmer, U. (1995), Adaptive Approaches to Basic Mobile Robot Tasks. PhD thesis, University of Kaiserslautern

Inductive Conformal Prediction: Theory and Application to Neural Networks

Harris Papadopoulos
Frederick University
Cyprus

1. Introduction

Traditional machine learning algorithms for pattern recognition just output *simple predictions*, without any associated *confidence values*. Confidence values are an indication of how likely each prediction is of being correct. In the ideal case, a confidence of 99% or higher for all examples in a set, means that the percentage of erroneous predictions in that set will not exceed 1%.

Knowing the likelihood of each prediction enables us to assess the extent to which we can rely on it. For this reason, predictions that are associated with some kind of confidence values are highly desirable in many risk-sensitive applications, such as those used for medical diagnosis or financial analysis. In fact, such information can benefit any application that requires human-computer interaction, as confidence values can be used to determine the way in which each prediction will be treated. For instance, a filtering mechanism can be employed so that only predictions which satisfy a certain level of confidence will be taken into account, while the rest can be discarded or passed on to a human for judgement.

There are two main areas in mainstream machine learning that can be used in order to obtain some kind of confidence values; the Bayesian framework and the theory of Probably Approximately Correct learning (PAC theory). Quite often the Bayesian framework is used for producing algorithms that complement individual predictions with probabilistic measures of their quality. On the other hand, PAC theory can be used for producing upper bounds on the probability of error for a given algorithm with respect to some confidence level $1 - \delta$. Both of these approaches however, have their drawbacks.

In order to apply the Bayesian framework one is required to have some prior knowledge about the distribution that generates the data. When the correct prior is known, Bayesian methods provide optimal decisions. For real world data sets though, as the required knowledge is not available, one has to assume the existence of an arbitrarily chosen prior. In this case, if the assumed prior is incorrect, the resulting confidence levels may also be "incorrect"; for example the predictive regions output for the 95% confidence level may contain the true label in much less than 95% of the cases. This signifies a major failure, as we would expect confidence levels to bound the percentage of expected errors. An experimental demonstration of how misleading Bayesian methods can become when their assumptions are violated can be found in (Melluish et al., 2001).

PAC theory on the contrary, only assumes that the data are generated by some completely unknown i.i.d. distribution. There are some PAC methods that are capable of establishing non-trivial bounds that might be interesting in practice. In order for them to do so though, the data set should be particularly clean. If this is not the case, which is not for the majority of data sets, the bounds obtained from these methods are very loose and as such they are not very useful in practice. A demonstration of the crudeness of PAC bounds can be found in (Nouretdinov et al., 2001a), where there is an example of Littlestone and Warmuth's bound (found in (Cristianini & Shawe-Taylor, 2000), Theorems 4.25 and 6.8) applied to the USPS data set. In addition, PAC theory has two other drawbacks: (a) the majority of relevant results either involve large explicit constants or do not specify the relevant constants at all; (b) the bounds obtained by PAC theory are for the overall error and not for individual test examples.

A new approach to obtaining confidence values was suggested in (Saunders et al., 1999) and (Vovk et al., 1999), where what we call in this chapter "Conformal Prediction" (CP) was proposed. Conformal Predictors are built on top of traditional algorithms, called underlying algorithms, but unlike the latter they complement each of their predictions with a measure of confidence; they also produce a measure of "credibility", which serves as an indicator of how suitable the training data are for classifying the example.

In contrast to Bayesian methods, CPs give probabilistically valid results, as they are only based on the general i.i.d. assumption; a comparison between Bayesian methods and CPs can be found in (Melluish et al., 2001). Furthermore, unlike PAC theory, they produce confidence measures that are useful in practice and are associated with individual test examples. Different variants of CPs are described in the papers (Saunders et al., 1999; Nouretdinov et al., 2001b; Proedrou et al., 2002; Papadopoulos et al., 2008). The results reported in these papers show that not only the confidence values output by CPs are useful in practice, but also that their accuracy is comparable to, and sometimes even better than, that of traditional machine learning algorithms.

The only disadvantage of CPs is their relative computational inefficiency. This is due to the use of transductive inference, since all computations have to start from scratch for every test example. Unfortunately, this computational inefficiency problem renders the application CP highly unsuitable for any approach that requires long training times such as Neural Networks. For this reason, a modification of the original CP approach called "Inductive Conformal Prediction" (ICP) was proposed in (Papadopoulos et al., 2002a) for regression and in (Papadopoulos et al., 2002b) for pattern recognition. As suggested by its name, ICP replaces the transductive inference used in the original approach with inductive inference. As a result, ICPs are almost as computationally efficient as their underlying algorithms.

This chapter gives a detailed study of ICP and describes its application to Neural Networks, which is one of the most widely used approaches for solving machine learning problems. The next Section summarises the general idea of Conformal Prediction, while Section 3 details the way CPs and ICPs work and analyses the impact that the choice between trunsductive and inductive inference has on the performance of Conformal Prediction. Note that in order to differentiate clearly between the original CP and ICP approaches, the former will be mostly called Transductive Conformal Prediction (TCP). Section 4 explains the application of ICP to Neural Networks, first presented in (Papadopoulos et al., 2007), and Section 5 lists some experimental results obtained by testing the Neural Networks ICP on benchmark data sets. Finally, Section 6 presents the conclusions of the chapter.

2. Conformal prediction

This Section gives an outline of the main idea behind conformal prediction; for more details see (Vovk et al., 2005). We are given a training set ($z_1, \dots, z_l$) of examples, where each $z_i \in Z$ is a pair (x_i, y_i); $x_i \in \mathbb{R}^d$ is the vector of attributes for example i and y_i is the classification for that example. We are also given a new unclassified example x_{l+1} and our task is to predict the classification y_{l+1} of this example. We know a priori the set of all possible labels $Y_1, \dots, Y_c$ and our only assumption, as with all the problems we are interested in, is the general i.i.d. model ($z_1, z_2, \dots$ are independent and identically distributed). Now suppose we can measure how likely it is that a given sequence of classified examples were drawn independently from the same probability distribution; in other words how typical the sequence is wrt the i.i.d. model. Then by measuring the typicalness of the extended sequence

$$((x_1, y_1), \dots, (x_l, y_l), (x_{l+1}, Y_j)) \tag{1}$$

we would in effect be measuring the likelihood of the label Y_j being the true label of our new example x_{l+1}, since this is the only component of our sequence that was not given to us. In the spirit of (Martin-Löf, 1966), a function p: $Z^* \to [0,1]$ is a test for randomness wrt the i.i.d. model if

- $\forall n \in \mathbb{N}$, $\forall \delta \in [0,1]$ and for all probability distributions P on Z,

$$P^n\{z \in Z^n : p(z) \le \delta\} \le \delta; \tag{2}$$

- p is semi-computable from above.

We will use the term *p-value function* to refer to such a function, since this definition is practically equivalent to the notion of p-values used in traditional statistics. In effect, the second requirement, that p is semi-computable from above, is completely irrelevant from the practical point of view, since the p-value functions of any interest in applications of statistics are always computable.

Therefore, we can obtain the typicalness of a sequence of examples by using a computable function p: $Z^* \to [0,1]$ which satisfies (2). We will call the output of this function for the sequence

$$((x_1, y_1), \dots, (x_l, y_l), (x_{l+1}, Y_j)) \tag{3}$$

(where Y_j is one of the c possible labels of our new example) the p-value of Y_j and denote it by $p(Y_j)$. If the p-value of a given label is under some very low threshold, say 0.05, this would mean that this label is highly unlikely, since such sequences will only be generated at most 5% of the time by any i.i.d. process.

A p-value function can be constructed by considering how different each example in our sequence is from all other examples. In order to formalize the fact that the order in which examples appear should not matter we use the concept of a bag (also called a multiset); we write $\llbracket z_1, \dots, z_n \rrbracket$ to denote the bag consisting of the elements $z_1, \dots, z_n$. We use a family of functions $A_n : Z^{(n-1)} \times Z \to \mathbb{R}$, $n = 1, 2, \dots$, which assign a numerical score

$$\alpha_i = A_n(\llbracket z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_n \rrbracket, z_i), \tag{4}$$

to each example z_i, indicating how different it is from the examples in the bag $\llbracket z_1,\dots,z_{i-1},z_{i+1},\dots,z_n \rrbracket$; such families of functions are called *nonconformity measures*. The nonconformity scores of all examples can now be used for computing the p-value of our sequence with the function

$$p(z_1,\dots,z_n) = \frac{\#\{i=1,\dots,n : \alpha_i \geq \alpha_n\}}{n}. \tag{5}$$

This function satisfies (2); a proof can be found in (Nouretdinov et al., 2001a).

2.1 Measuring nonconformity

We can measure the nonconformity α_i of each example z_i in a bag $\llbracket z_1,\dots,z_n \rrbracket$ with the aid of some traditional machine learning method, which we call the *underlying algorithm* of the CP. Given a bag of examples $\llbracket z_1,\dots,z_n \rrbracket$ as training set, each such method creates a prediction rule $D_{\llbracket z_1,\dots,z_n \rrbracket}$, which maps any unclassified example x to a label $\hat{y}$. As this prediction rule is based on the examples in the bag, the deviation of the predicted label

$$\hat{y}_i = D_{\llbracket z_1,\dots,z_n \rrbracket}(x_i) \tag{6}$$

from the actual label y_i of the example z_i tells us how different z_i is from the rest of the examples in the bag. Therefore, this deviation gives us a measure of the nonconformity of example z_i.

Alternatively, we can create the prediction rule $D_{\llbracket z_1,\dots,z_{i-1},z_{i+1},\dots,z_n \rrbracket}$ using all the examples in the bag except z_i, and measure the deviation of

$$\hat{y}_i = D_{\llbracket z_1,\dots,z_{i-1},z_{i+1},\dots,z_n \rrbracket}(x_i) \tag{7}$$

from y_i.

3. Transductive and inductive conformal predictors

This Section gives a general description of the way Transductive and Inductive Conformal Predictors work and analyses their differences. Before focusing on Conformal Predictors, it first looks at the main concepts of Transductive and Inductive inference. It then details the steps that the two approaches follow and explains the meaning of the confidence and credibility measures they produce. This is followed by the presentation of an alternative mode in which CPs can be used and a discussion about the difference between the validity and the usefulness of their results. Finally, it compares the two approaches both in terms of computational efficiency and accuracy.

3.1 Transductive and inductive inference

The difference between transductive and inductive inference is very informal and far from being clear-cut. In this subsection we describe the main idea behind each one, so that the difference between Transductive and Inductive Conformal Prediction becomes clearer.

In inductive inference, we use our training set to generate a more or less general "rule" (or "model", or "theory") about the data, which we then apply to each test pattern to obtain our predictions. Hence all the information we need from the training set are incorporated into our general "rule" and we do not make any direct use of the training examples in order to produce each prediction. In Transductive inference on the other hand, the first step, generating a general "rule", is skipped. Thus no processing is applied to the training set beforehand and all our computations are based on each individual test example, using the actual training set for deriving our prediction.

3.2 Transductive conformal prediction

The general steps the Transductive Conformal Prediction approach follows for a given input vector x_{l+g} are:

- Consider all possible classifications $Y_1, \dots, Y_c$ and apply the underlying algorithm to every one of the possible completions

$$\begin{array}{c}(x_1,y_1),\dots,(x_l,y_l),(x_{l+g},Y_1)\\ \vdots \\ (x_1,y_1),\dots,(x_l,y_l),(x_{l+g},Y_c)\end{array} \tag{8}$$

- For every possible completion Y_j, assign a nonconformity score to each training example $(x_1, y_1), \dots, (x_l, y_l)$ and to the pair (x_{l+g}, Y_j). This process will result in the sequences

$$\begin{array}{c}\alpha_1^{(Y_1)},\dots,\alpha_l^{(Y_1)},\alpha_{l+g}^{(Y_1)}\\ \vdots \\ \alpha_1^{(Y_c)},\dots,\alpha_l^{(Y_c)},\alpha_{l+g}^{(Y_c)}\end{array} \tag{9}$$

- Compute the p-value for x_{l+g} being classified as each possible label Y_j by applying (5) to the corresponding sequence

$$\alpha_1^{(Y_j)},\dots,\alpha_l^{(Y_j)},\alpha_{l+g}^{(Y_j)}. \tag{10}$$

  So that

$$p(Y_j)=\frac{\#\{i=1,\dots,l,l+g:\alpha_i^{(Y_j)}\geq\alpha_{l+g}^{(Y_j)}\}}{l+1}. \tag{11}$$

- Predict the classification with the largest p-value.
- Output as confidence to this prediction one minus the second largest p-value, and as credibility the p-value of the output prediction, i.e. the largest p-value.

Note that this process, which includes the application of the CPs underlying algorithm c times, is repeated for each input vector $x_{l+1},\dots, x_{l+r}$. This is the cause of the computational inefficiency of the original CP approach.

3.3 Inductive conformal prediction

Inductive Conformal Prediction splits the training set into two parts, the *proper training set* and the *calibration set*. It then applies the underlying algorithm to the proper training set and uses the examples in the calibration set together with the new example to compute the p-value for each possible classification. As a result, it only needs to apply the underlying algorithm once. The general steps inductive conformal prediction follows are:

- Split the training set into two smaller sets, the proper training set with $m := l - q$ examples and the calibration set with q examples; where q is a parameter of the algorithm.
- Use the proper training set $(z_1,\dots,z_m)$ to generate a general rule $D_{[\![z_1,\dots,z_m]\!]}$ for classifying new examples; this general rule is created by the underlying algorithm.
- Assign a nonconformity score to each one of the examples in the calibration set. This will result in the sequence $\alpha_{m+1},\dots,\alpha_{m+q}$.
- For each input vector $x_{l+g} : g = 1,\dots,r$,
  - consider each possible classification $Y_j : j = 1,\dots,c$ and compute the nonconformity score $\alpha_{l+g}^{(Y_j)}$ of each pair (x_{l+g}, Y_j).
  - Compute the p-value for x_{l+g} being classified as each possible label Y_j by applying (5) to the nonconformity scores of the calibration examples together with $\alpha_{l+g}^{(Y_j)}$:

$$\alpha_{m+1},\dots,\alpha_{m+q},\alpha_{l+g}^{(Y_j)}. \tag{12}$$

  So that

$$p(Y_j) = \frac{\#\{i = m+1,\dots,m+q,l+g : \alpha_i \geq \alpha_{l+g}^{(Y_j)}\}}{q+1}. \tag{13}$$

  - Predict the classification with the largest p-value.
  - Output as confidence to this prediction one minus the second largest p-value, and as credibility the p-value of the output prediction, i.e. the largest p-value.

The parameter q in the above scheme determines the number of training examples that will be allocated to the calibration set and the nonconformity scores of which will be used by the ICP to calculate its p-values. These examples should only take up a small portion of the training set, so that their removal will not dramatically reduce the predictive ability of the underlying algorithm. As we are mainly interested in the confidence levels of 99% and 95%, the calibration sizes we use are of the form $q = 100n - 1$, where $n \in \mathbb{N}$; according to (13), in order for a prediction to have a confidence level of 99%, the new example should be among the $\left\lfloor \frac{q+1}{100} \right\rfloor$ strangest examples when assigned all other possible classifications (so that the p-value of these classifications will be 0.01).

3.4 Confidence and credibility measures

Confidence gives us a measure of how likely our prediction is compared to all other possible classifications, according to the training set. To give a more detailed explanation of the confidence measure we need to recall the basic property of valid p-values; for any i.i.d. distribution P and for every significance level δ,

$$P\{p(Y_j) \leq \delta\} \leq \delta. \tag{14}$$

This means that if the p-value of a label Y_j is smaller than or equal to a significance level δ, then either Y_j is not the true label or an event of at most δ probability occurred. Now suppose that δ is equal to the second largest p-value for a given input vector x_{l+g} with a set of possible labels $\{Y_1, \ldots, Y_c\}$. This would mean that the probability of any label other than Y_j, where $p(Y_j)$ is the largest p-value, being the true label is at most δ. Consequently, our confidence value for each prediction is one minus the probability of any one of the other labels being the true label; the higher the confidence value for a prediction the less likely for it not being the true label. Note that this is an informal argument, as it uses a δ dependent on the observed examples.

The credibility measure is equal to the highest p-value of any one of the possible classifications being the true label according to the training set. As such it gives us an indication of how good the training set is for classifying the current example i.e. if the credibility of a prediction is very low this means that either the training set is not random or the new example is not representative of the training set.

3.5 The two modes of conformal prediction

The p-values obtained by (11) and (13) for each possible classification, can be used in two different modes:

- For each test example output the predicted classification together with a confidence and credibility measure for that classification.
- Given a confidence level $1 - \delta$, where $\delta > 0$ is the significance level (typically a small constant), output the appropriate set of classifications such that one can be $1 - \delta$ confident that the true label will be included in that set.

The first case corresponds to the algorithms detailed in Sections 3.2 and 3.3. In the second case the CP outputs the set

$$\{Y_j : p(Y_j) > \delta\}, \tag{15}$$

where $j = 1, \ldots, c$ (c is the number of possible classifications). In other words, it outputs the set consisting of all the classifications that have a greater than δ p-value of being the true label.

3.6 Validity and usefulness

The p-values computed by the functions (11) and (13) are valid in the sense of satisfying (2), provided that the data in question are drawn independently from the same probability distribution. Thus, the particular nonconformity measure definition and underlying algorithm used by a given conformal predictor do not influence the validity of its p-values in any way; i.e. if the model generating the data is i.i.d., the results produced by any ICP or TCP will be valid. In fact, any function can be used as a nonconformity measure definition, even if its output has nothing to do with how nonconforming the input example is. The use of such a measure will not have an impact on the validity of the results produced by the CP, but it will, on the other hand, affect their usefulness.

To demonstrate the influence of an inadequate nonconformity measure definition on the results of a CP, let us consider the case of a trivial definition that always returns the value of 1 for any given example. This will make the p-values of all possible labels equal to 1 and will result in a randomly chosen prediction with a confidence of 0%, which although is valid, does not provide us with any information. Therefore, if the nonconformity measure definition or the underlying algorithm of a CP are not suitable for the data in question, this will be reflected in the usefulness of the resulting confidence measures.

3.7 Comparison

Although the main motivation behind the introduction of transductive inference, by Vapnik (Vapnik, 1998), was the creation of more computationally efficient versions of learning algorithms, this does not seem to be the case in the theory of conformal prediction. The Transductive CP starts all computations from scratch each time a new unclassified example arrives. This turns out to be very computationally inefficient, since it means that for every test example it has to apply the underlying algorithm and compute all the nonconformity scores of the examples *c* times, one for each possible classification. On the other hand, the Inductive CP carries out all these computations in advance, which makes it much faster. It only needs to calculate the nonconformity scores of the new example being assigned each one of the possible classifications, using the already generated general rule.

In order to analyse further the computational efficiency difference between ICP and TCP, let us consider the computational complexity of each method with respect to the complexity of its underlying algorithm *U*. The complexity of *U* when applied to a data set with *l* training examples and *r* test examples will be

$$\Theta(U_{\text{train}}(l) + rU_{\text{apply}}), \tag{16}$$

where $U_{\text{train}}(l)$ is the time required by *U* to generate its general rule and U_{apply} is the time needed to apply this general rule to a new example. Note that although the complexity of any algorithm also depends on the number of attributes *d* that describe each example, this was not included in our notation for simplicity reasons. The corresponding complexity of the TCP will be

$$\Theta(rc(U_{\text{train}}(l+1) + (l+1)U_{\text{apply}})), \tag{17}$$

where *c* is the number of possible labels of the task; we assume that the computation of the nonconformity scores and p-values for each possible label is relatively fast compared to the time it takes to train and apply the underlying algorithm. Analogously, the complexity of the ICP will be

$$\Theta(U_{\text{train}}(l-q) + (q+r)U_{\text{apply}}), \tag{18}$$

where *q* is the size of the calibration set. Notice that the ICP takes less time than the original method to generate the prediction rule, since

$$U_{\text{train}}(l-q) < U_{\text{train}}(l), \tag{19}$$

while it then repeats U_{apply} a somewhat larger amount of times. The time needed for applying the prediction rule of most inductive algorithms, however, is insignificant compared to the amount of time spent for generating it. Consequently, the ICP will in most cases be slightly faster than the original method, as it spends less time during the most complex part of its underlying algorithm's computations. On the contrary, the corresponding TCP repeats a slightly bigger number of computations than the total computations of its underlying algorithm for *rc* times, thing that makes it much slower than both the original method and the ICP.

The only drawback of the ICP is a small loss in terms of accuracy. This is due to the fact that the TCP uses all the training examples for the training of its underlying algorithm, whereas the ICP uses only the examples in the proper training set. Furthermore, the TCP uses a richer set of nonconformity scores, computed from all the training examples, when calculating the p-values for each possible classification, as opposed to the small part of training examples, the calibration set, the ICP uses for the same purpose. As the experimental results in Section 5 and in (Papadopoulos et al., 2002a; Papadopoulos et al., 2002b; Papadopoulos et al., 2007) show, this loss of accuracy is negligible while the improvement in computational efficiency is massive.

4. Neural networks ICP

In this Section we analyse the Neural Networks ICP. This method can be implemented in conjunction with any Neural Network for pattern recognition as long as it uses the 1-of-*n* output encoding, which is the typical encoding used for such networks. We first give a detailed description of this encoding and then move on to the definition of two nonconformity measures for Neural Networks. Finally, we detail the Neural Networks ICP algorithm.

4.1 Output encoding

Typically the output layer of a classification Neural Network consists of *c* units, each representing one of the *c* possible classifications of the problem at hand; thus each label is encoded into *c* target outputs. To explicitly describe this encoding consider the label, $y_i = Y_u$ of a training example *i*, where $Y_u \in \{Y_1, \dots, Y_c\}$ is one of the *c* possible classifications. The resulting target outputs for y_i will be

$$t_1^i, \dots, t_c^i \tag{20}$$

where

$$t_j^i = \begin{cases} 1, \text{ if } j = u, \\ 0, \text{ otherwise} \end{cases} \tag{21}$$

for $j = 1, \dots, c$. Here we assumed that the Neural Network in question has a softmax output layer, as this was the case for the networks used in our experiments. The values 0 and 1 can be adjusted accordingly depending on the range of the output activation functions of the network being used.

As a result of this encoding, the prediction $\hat{y}_g$ of the network, for a test pattern *g*, will be the label corresponding to its highest output value.

4.2 Nonconformity measures

According to the above encoding, for an example *i* with true classification Y_u, the higher the output o_u^i (which corresponds to that classification) the more conforming the example, and the higher the other outputs the less conforming the example. In fact, the most important of all other outputs is the one with the maximum value $\max_{j=1,\dots,c:j\neq u} o_j^i$, since that is the one which might be very near or even higher than o_u^i.

So a natural nonconformity measure for an example $z_i = (x_i, y_i)$ where $y_i = Y_u$ would be defined as

$$\alpha_i = \max_{j=1,\ldots,c:j\neq u} o_j^i - o_u^i, \tag{22}$$

or as

$$\alpha_i = \frac{\max_{j=1,\ldots,c:j\neq u} o_j^i}{o_u^i + \gamma}, \tag{23}$$

where the parameter $\gamma \geq 0$ in the second definition enables us to adjust the sensitivity of our measure to small changes of o_u^i depending on the data in question. We added this parameter in order to gain control over which category of outputs will be more important in determining the resulting nonconformity scores; by increasing γ one reduces the importance of o_u^i and consequently increases the importance of all other outputs.

4.3 The algorithm

We can now follow the general ICP algorithm detailed in Section 3.3 together with nonconformity measure (22) or (23) to produce the Neural Network ICP. More specifically, the exact steps the Neural Network ICP follows are:

- Split the training set into two smaller sets, the proper training set with $m := l - q$ examples and the calibration set with q examples; where q is a parameter of the algorithm.
- Use the proper training set to train the neural network.
- For each example $z_{m+t} = (x_{m+t}, y_{m+t}): t = 1,\ldots,q$ in the calibration set,
  - supply the input pattern x_{m+t} to the trained network to obtain the output values $o_1^{m+t},\ldots,o_c^{m+t}$ and
  - calculate the nonconformity score α_{m+t} of the pair (x_{m+t}, y_{m+t}) by applying (22) or (23) to these values.
- For each test pattern $x_{l+g} : g = 1,\ldots,r$,
  - supply the input pattern x_{l+g} to the trained network to obtain the output values $o_1^{l+g},\ldots,o_c^{l+g}$,
  - consider each possible classification $Y_u : u = 1,\ldots,c$ and
    - compute the nonconformity score $\alpha_{l+g}^{(Y_u)}$ of the pair (x_{l+g}, Y_u) by applying (22) or (23) to the outputs of the network,
    - calculate the p-value $p(Y_u)$ for x_{l+g} being classified as Y_u by applying (5) to the nonconformity scores of the calibration examples and $\alpha_{l+g}^{(Y_u)}$:

$$p(Y_u) = \frac{\#\{i = m+1,\ldots,m+q, l+g : \alpha_i \geq \alpha_{l+g}^{(Y_u)}\}}{q+1}, \tag{24}$$

  - predict the classification with the smallest nonconformity score,
  - output as confidence to this prediction one minus the second largest p-value, and as credibility the p-value of the output prediction, i.e. the largest p-value.

5. Experimental results of the neural networks ICP

Here we detail the experimental results of the Neural Networks ICP on the Satellite, Shuttle and Segment data sets, which were used in the experiments of the Statlog Project (King et al., 1995); see also (Michie et al., 1994).

The Satellite data set consists of 6435 satellite images, split into 4435 training examples and 2000 test examples, described by 36 attributes. The classification task is to distinguish among 6 different soil conditions that are represented in the images. The calibration set was formed from 199 of the 4435 training examples.

The Shuttle data set consists of 43500 training examples and 14500 test examples with 9 attributes each, describing the conditions in a space shuttle. The classification task is to choose which one of the 7 different sets of actions should be taken according to the conditions. In this case we used 999 of the training examples to form the calibration set.

The Segment data set consists of 2310 outdoor images described by 18 attributes each. The classification task is to choose between: brick-face, sky, foliage, cement, window, path, grass. For our experiments on this data set we used 10 fold cross-validation, as this was the testing procedure followed in the Statlog project. The set was divided into 10 equally sized parts and our tests were repeated 10 times, each time using one of the 10 parts as the test set and the remaining 9 as the training set. Consequently, the resulting training and test sets consisted of 2079 and 231 examples respectively. Of the 2079 training examples 199 were used to form the calibration set.

Our experiments on these data sets were performed using 2 layer fully connected networks, with sigmoid hidden units and softmax output units. The number of their input and output units were determined by the format of each data set; equal to the number of attributes and possible classifications of the examples respectively. These networks were trained with the backpropagation algorithm minimizing a cross-entropy loss function. The number of hidden units and the learning and momentum rates used for each data set are reported in table 1. It is worth to note that the same parameters were used both for the ICP and its underlying algorithm.

Here we report the error percentages of the Neural Network ICP and compare them to the ones of its underlying algorithm as well as to those of some other traditional methods. In addition, we check the quality of its p-values by analysing the results obtained from its second mode, described in Section 3.5, for the 99%, 95% and 90% confidence levels. For the purpose of reporting the results of this mode we separate its outputs into three categories:

- A set with more than one labels
- A set with only one label
- The empty set

Our main concern here will be the number of outputs that belong to the first category; we want this number to be relatively small, since these are the examples for which the ICP is not certain in only one label at the required confidence level $1 - \delta$. In addition to the percentage of examples in each category, we also report the number of errors made by the ICP in this mode. This is the number of examples for which the true label was not included in the set output by the ICP; including all cases where the set output by the ICP was empty. Over many runs on different sets (both training and test) generated from the same i.i.d. distribution, the percentage of these errors will be close to the corresponding significance level δ; an experimental demonstration of this can be found in (Vovk, 2002). Finally, we examine the computational efficiency of the method by comparing its processing times with those of its underlying algorithm.

	Satellite	Shuttle	Segment
Hidden Units	23	12	11
Hidden Learning Rate	0.002	0.002	0.002
Output Learning Rate	0.001	0.001	0.001
Momentum Rate	0.1	0	0.1

Table 1. The parameters used in our experiments for each data set.

Learning Algorithm	Percentage of error (%)		
	Satellite	Shuttle	Segment
Neural Networks ICP	10.40	0.0414	3.46
Backpropagation	10.24	0.0414	3.20
k-Nearest Neighbours	9.45	0.12	3.68
C4.5	15.00	0.10	4.00
CART	13.80	0.08	4.00
Naïve Bayes	28.70	4.50	26.50
CASTLE	19.40	3.80	11.20
Linear Discriminant	17.10	4.83	11.60

Table 2. Error rate comparison of the Neural Networks ICP with traditional algorithms.

In table 2 we compare the performance of the ICP on the three statlog project data sets with that of its underlying algorithm (we denote this as backpropagation) and that of 6 other traditional methods. These are the *k*-Nearest Neighbours algorithm, two decision tree algorithms, namely C4.5 and CART, the Naïve Bayes classifier, a Bayesian network algorithm called CASTLE and a linear discriminant algorithm. The results of the *k*-Nearest Neighbours algorithm were produced by the author, while those of all other methods were reported in (King et al., 1995) and (Michie et al., 1994). Note that the aim of the CP is not to outperform other algorithms but to produce more information with each prediction. So in comparing these error percentages we want to show that the accuracy of this method is not inferior to that of traditional algorithms.

We did not perform the same experiments with the corresponding original CP algorithm, due to the huge amount of time that would have been needed for doing so. However, its results in terms of error percentages would not have been significantly different from those of its underlying algorithm (backpropagation). So the first two rows of table 2 also serve as a performance comparison between ICP and TCP.

Table 2 clearly shows that the accuracy of the Neural Networks ICP is comparable to that of traditional methods. Of course, our main comparison here is with the performance of its underlying algorithm, since that is where ICPs base their predictions and since that is also the performance of the corresponding TCP. So by comparing its results to those of the backpropagation method, we can see that although in most cases the ICP suffers a small loss of accuracy, this loss is negligible. Moreover, we observe that as the data set gets bigger the difference between the error percentage of the ICP and that of its underlying algorithm

becomes smaller. In fact, for the shuttle data set, which is the biggest, the ICP gives exactly the same results with its underlying network.
Tables 3 to 5 detail the performance of the second mode of the ICP on each of the three data sets. Here we can see that the percentage of examples for which it needs to output more than one label is relatively small even for a confidence level as high as 99%, having in mind the difficulty of each task and the performance of its underlying algorithm on each data set. This reflects the quality of the p-values calculated by this method and consequently the usefulness of its confidence measures.

Nonconformity Measure	Confidence Level	Only one Label (%)	More than one label (%)	No Label (%)	Errors (%)
(4)	99%	60.72	39.28	0.00	1.11
	95%	84.42	15.58	0.00	4.67
	90%	96.16	3.02	0.82	9.59
(5)	99%	61.69	38.31	0.00	1.10
	95%	85.70	14.30	0.00	4.86
	90%	96.11	3.10	0.79	9.43

Table 3. Results of the second mode of the Neural Networks ICP for the Satellite data set.

Nonconformity Measure	Confidence Level	Only one Label (%)	More than one label (%)	No Label (%)	Errors (%)
(4)	99%	99.23	0.00	0.77	0.77
	95%	93.52	0.00	6.48	6.48
	90%	89.08	0.00	10.92	10.92
(5)	99%	99.30	0.00	0.70	0.70
	95%	93.86	0.00	6.14	6.14
	90%	88.72	0.00	11.28	11.28

Table 4. Results of the second mode of the Neural Networks ICP for the Shuttle data set.

Nonconformity Measure	Confidence Level	Only one Label (%)	More than one label (%)	No Label (%)	Errors (%)
(4)	99%	90.69	9.31	0.00	0.95
	95%	97.71	1.25	1.04	3.68
	90%	94.68	0.00	5.32	6.71
(5)	99%	91.73	8.27	0.00	1.04
	95%	97.79	1.21	1.00	3.55
	90%	94.76	0.00	5.24	6.67

Table 5. Results of the second mode of the Neural Networks ICP for the Segment data set.

Finally, table 6 lists the processing times of the Neural Network ICP together with those of its underlying algorithm. In the case of the Segment data set the times listed are for the total duration of the experiments on all 10 splits. As mentioned in the computational complexity comparison of Section 3.7, in most cases the ICP is faster than its underlying algorithm because it uses less training examples. In the case of Neural Networks, this reduction in training examples reduces slightly the training time per epoch and, for more or less the

same number of epochs, it results in a shorter total training time. This was the case for the Satellite and Shuttle data sets. However, for the Segment data set the number of epochs increased and this resulted in a slightly bigger total training time for the ICP.
Based on our computational complexity analysis of Section 3.7, if we were to perform the same experiments using the original CP method coupled with Neural Networks it would have taken approximately 183 days for the Satellite data set, 53 years for the Shuttle data set and 93 days for the Segment data set. This shows the huge computational efficiency improvement of ICP in the case of Neural Networks. In fact, it shows that ICP is the only conformal prediction method that can be used with this approach.

Learning Algorithm	Time (in seconds)		
	Satellite	Shuttle	Segment
Neural Networks ICP	1077	11418	5322
Backpropagation	1321	16569	4982

Table 6. The processing times of the Neural Networks ICP and its underlying algorithm.

6. Conclusion

This chapter presented the Inductive Conformal Prediction (ICP) approach for producing confidence measures with predictions and described its application to Neural Networks. ICPs accompany each of their predictions with probabilistically valid measures of confidence. Furthermore, they do not need the relatively large amount of processing time spend by Transductive Conformal Predictors (TCPs) to perform their computations. In fact their computational efficiency is virtually the same with that of their underlying algorithms.
The experimental results detailed in Section 5 and in (Papadopoulos et al., 2002a; Papadopoulos et al., 2002b; Papadopoulos et al., 2007) show that the accuracy of ICPs is comparable to that of traditional methods, while the confidence measures they produce are useful in practice. Of course, as a result of removing some examples from the training set to form the calibration set, they sometimes suffer a small, but usually negligible, loss of accuracy from their underlying algorithm. This is not the case, however, for large data sets, which contain enough training examples so that the removal of the calibration examples does not make any difference to the training of the algorithm.

7. Acknowledgements

This work was supported by the Cyprus Research Promotion Foundation through research contract PLHRO/0506/22 ("Development of New Conformal Prediction Methods with Applications in Medical Diagnosis").

8. References

Cristianini, N. & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Methods*, Cambridge University Press, Cambridge.

King, R.; Feng, C. & Sutherland, A. (1995). Statlog: Comparison of classification algorithms on large real-world problems, *Applied Artificial Intelligence*, Vol. 9, No. 3, pp. 259–287.

Martin-Löf, P. (1966). The definition of random sequences. *Information and Control*, Vol. 9, pp. 602–619.

Melluish, T.; Saunders, C.; Nouretdinov, I. & Vovk, V. (2001). Comparing the Bayes and Typicalness frameworks, *Proceedings of the 12th European Conference on Machine Learning (ECML'01)*, Vol. 2167 of *Lecture Notes in Computer Science*, pp. 360–371, Springer.

Michie, D.; Spiegelhalter, D. & Taylor, C. (Ed.) (1994). *Machine Learning, Neural and Statistical Classification*, Ellis Horwood.

Nouretdinov, I.; Vovk, V.; Vyugin, M. & Gammerman, A. (2001a). Pattern recognition and density estimation under the general iid assumption, *Proceedings of the 14th Annual Conference on Computational Learning Theory (COLT'01) and 5th European Conference on Computational Learning Theory (EuroCOLT'01)*, Vol. 2111 of *Lecture Notes in Computer Science*, pp. 337–353, Springer.

Nouretdinov, I.; Melluish, T. & Vovk. V. (2001b). Ridge regression confidence machine, *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*, pp. 385–392, Morgan Kaufmann, San Francisco, CA.

Papadopoulos, H.; Proedrou, K.; Vovk, V. & Gammerman, A. (2002a). Inductive confidence machines for regression, *Proceedings of the 13th European Conference on Machine Learning (ECML'02)*, Vol. 2430 of *Lecture Notes in Computer Science*, pp. 345–356, Springer.

Papadopoulos, H.; Vovk, V. & Gammerman, A. (2002b). Qualified predictions for large data sets in the case of pattern recognition, *Proceedings of the 2002 International Conference on Machine Learning and Applications (ICMLA'02)*, pp. 159–163, CSREA Press.

Papadopoulos, H.; Vovk, V. & Gammerman, A. (2007). Conformal prediction with neural networks, *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'07)*, Vol. 2, pp. 388-395, Patras, Greece, October 2007, IEEE Computer Society, Los Alamitos, CA.

Papadopoulos, H.; Gammerman, A. & Vovk, V. (2008). Normalized nonconformity measures for regression conformal prediction, Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2008), pp. 64-69, Innsbruck, Austria, February 2008, ACTA Press.

Proedrou, K.; Nouretdinov, I.; Vovk, V. & Gammerman, A. (2002). Transductive confidence machines for pattern recognition, *Proceedings of the 13th European Conference on Machine Learning (ECML'02)*, Vol. 2430 of *Lecture Notes in Computer Science*, pp. 381–390. Springer.

Saunders, C.; Gammerman, A. & Vovk, V. (1999). Transduction with confidence and credibility, *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, Vol. 2, pp. 722–726, Morgan Kaufmann, Los Altos, CA.

Vapnik, V. (1998). *Statistical Learning Theory*, Wiley, New York.

Vovk, V.; Gammerman, A. & Saunders, C. (1999). Machine learning applications of algorithmic randomness, *Proceedings of the 16th International Conference on Machine Learning (ICML'99)*, pp. 444–453, Morgan Kaufmann, San Francisco, CA.

Vovk, V. (2002). On-line confidence machines are well-calibrated, *Proceedings of the 43rd Annual Symposium on Foundations of Computer Science (FOCS'02)*, pp. 187–196, IEEE Computer Society, Los Alamitos, CA.

Vovk, V.; Gammerman, A. & Shafer, G. (2005). *Algorithmic Learning in a Random World*, Springer, New York.

Robust Classification of Texture Images using Distributional-based Multivariate Analysis

Vasileios K. Pothos, Christos Theoharatos, George Economou and Spiros Fotopoulos
Electronics laboratory, Dept. of Physics, University of Patras Greece

1. Introduction

Classification of texture images has been recognized as an important task in the field of image analysis and computer vision through the last few decades. A plethora of research papers have appeared in the literature trying to cope with effective ways to extract faithful distributions that accurately represent the inner content and attributes of texture images. An issue of great importance is, also, the incorporation of a valid similarity measure that can successfully estimate how close these distributions are with respect to some pre-classified texture categories. The basic operations that need to be carried out in order to estimate the similarity between texture images and thereafter assess the classification problem are (a) choose an appropriate feature space for texture representation, (b) construct a theoretically valid distribution in the texture feature space, i.e. the texture signature, which provide a representation of the texture image in a multivariate feature space, (c) perform pairwise comparisons between corresponding texture signatures that constitute the consequent content distributions of the texture images and (d) choose an experimentally valid classifier for the subsequent evaluation.
The scope of this chapter is the survey of a recently introduced methodology for distributional-based classification of texture images (Pothos et al., 2007), its enhancement via the incorporation of a self-organizing module and its adaptation so as to work in multivariate feature spaces. The original approach is based on an efficient strategy for analyzing texture patterns within a distributional framework and the use of a statistical distributional measure for comparing multivariate data, also known as the multivariate Wald-Wolfowitz test (WW-test) (Friedman & Rafsky, 1979). By combining the flexible character of the original methodology with the learning abilities of neural networks, we build a general-purpose platform for the efficient information management and classification of texture patches without any restriction regarding the exact image content.
Here, we will first describe the enrollment of standard feature extraction techniques for summarizing texture information and structuring multivariate texture spaces. These techniques include wavelet analysis, discrete cosine transform (DCT), Gabor filters and edge histogram descriptor. The above methods have been considered as golden standards for extracting appropriate distributions from texture images. In the following stage, we will test the applicability of some multivariate distributional-based measures for estimating the

classification accuracy over widely available databases and outline the different alternative facets under which a texture database can be accessed within the introduced platform. In this way, the multivariate distributions representing the individual images will be compared via the standard WW-test and the Kantorovich-Wasserstein (KWass) distance, building a content-based texture classification scheme. The proposed distributional measures will be revealed to handle efficiently the texture-space dimensionality and the limited sample size drawn from a given image.

In order to further boost the classification accuracy of the entire scheme, we will finally introduce a computational intelligent module for content representation based on a self-organizing neural network (SONN), the Neural-Gas algorithm (Martinez et al., 1993). The incorporated feature-extraction unit will be responsible for generating a parsimonious description of the texture distribution of each image. The resulting performance will be used to evaluate the four utilized approaches for texture representation. Emphasis will be given to the study of the two above subject and not in the design of the classifier.

2. Background and related work

Besides color and shape, texture plays an important role in the human visual system to recognize and categorize objects and properties in several kinds of images, from natural and artificial color images, to medical, remote sensing and quality control ones. It is proven to be an important visual property of the materials, encountered in many low-level image analysis and computer vision tasks. The study of texture is recognized to be a difficult subject in image science, while texture classification is a central research direction that has a wide variety of applications. In order to build a texture-based classification system, the basic building elements are first robust texture representation, then the design of a (dis)similarity measure between textures and finally the choice of the classifier.

Texture representation is a difficult problem due to the high – and usually unknown – true dimensionality of the feature space required to represent textures. Texture is defined as a homogeneous and coherent field of an image, characterized by features like roughness, variability, repeatability, directionality etc., which are characterized over a certain spatial extend. The preferred approach for texture feature extraction by the majority of researchers is based on image decomposition, by filtering with a subband or wavelet filter bank (Laine & Fan, 1993; Randen & Husoy, 1999; Leung & Malik, 2001; Do & Vetterli, 2002). The image is decomposed into several images for separate processing. The goal is to concentrate the involved energy in a few features and reduce the correlation. An analogous move is to apply a linear transformation by using Fourier or DCT. Older techniques that use direct image domain representation or express co-occurrence properties of image pixels are not commonly used lately. The most suitable representation for summarizing a nonparametric estimate of texture distribution is histogram, since texture is considered to describe the appearance of a region by the distribution of features rather than by some individual feature vectors (Ojala et al., 1996; Rubner et al., 2001). However, histogram is bound with the well-known binning problem which is a difficult one to solve. Regular binning of high-dimensional feature spaces results poor performance, coarse binning affects resolving power, while fine binning leads to large fluctuations due to the statistically insignificant sample sizes for most bins. In the literature, adaptive binning has been proposed to tackle the binning effect (Leow & Li, 2004), as well as binning induced by a set of prototypes.

The other important issue of a texture-based classification system is the design of an appropriate distance measure between textures. Towards this objective, several methods have been proposed based on histogram comparison. An alternative approach to measure texture resemblance is by means of non-parametric statistical tests that make no a-priori assumptions about the underlying sample distribution. This guarantees the similarities to be assessable in terms of statistical significance, but avoids direct statistical parameter estimation. Non-parametric and distributional based classification methods share a common characteristic; they require the availability of a number of independent – identically distributed – samples from the underlying distribution to operate on. These samples are extracted from the involved data during the texture representation stage. In the core of the proposed technique lies a non-parametric test dealing with the "Multivariate Two-Sample Problem", which has been adopted here for expressing texture image similarity. The specific test is a multivariate extension of the classical Wald-Wolfowitz test (WW-test) and compares two different vectorial samples by checking whether they form different branches in the overall minimal-spanning-tree (MST) (Friedman & Rafsky, 1979). The output of this test can be expressed as the probability that the two point-samples are coming from the same distribution. Its great advantage is that no a-priori assumption about the distribution of points in the two samples is a prerequisite. In order to enrich the evaluation results of texture image classification, another distributional distance is also utilized in this work to measure the dissimilarity between the extracted feature distributions, namely the Kantorovich-Wasserstein (KWass) distance (Gibbs & Su, 2002). It should be noticed that the KWass distance is equivalent to the well known Earth Movers Distance (EMD) (Rubner et al., 2001), which is an optimized solution to the transportation problem, when the later is applied on distributions with equal masses signatures.

Before examining in some detail the application of the above described methodology on a texture classification problem, it is interesting to discuss the importance of features' dimensions in an image classification problem. In general, individual image pixels are characterized by the grayscale value, which overall describe the image in a low-dimensionality feature space and can classify a set of images based on the first-order distribution of pixels' intensities. An image has many pixels and there are a large number of samples available to estimate the distribution. The texture case is quite different. For grayscale textures, instead of single pixels, a neighborhood needs to be considered as the basic texture distribution element, so as to account for pixel intensity correlations inside it. The size of the neighborhood is application dependent and should be large enough to encompass significant texture variation. Correlations among pixels can be accounted by increasing the space dimension required for sample representation, to equal the number of included pixels. Two are the basic implications of the above discussion; the expansion in space dimension that increases computational complexity and the limited sample size which in turn influences the classification error.

The success of the previously described methodologies for feature extraction and distributional similarity estimation were tested on a part of the OUTex and the Photometric texture database. The classification problem is stated as follows; given a new texture sample, assign it to the most similar one of a predetermined set of texture classes. Results should be in accordance with the intuitive notion of visual similarity of the different textures. Special effort is taken to judge all techniques under equal terms and use the available databases in an optimal way. Regarding texture feature extraction, four different methodologies were incorporated in this study which are considered as golden standard in the scientific community: wavelet transform, DCT, Gabor filters and edge histogram approach proposed in the MPEG-7 standard. These methodologies are shortly analyzed in the following section.

3. Texture feature extraction techniques

3.1 Wavelet transform

A compact representation of image texture needs to be derived in the transform domain for classification (Sebe & Lew, 2000). In the general case, the wavelet transform is applied to a given image in N decomposition levels, decomposing each level in four independent and spatially oriented channels, producing in this way the subbands LL, LH, HL and HH. For the texture feature extraction, the image is partitioned into M non-overlapping square blocks of specific size, and the wavelet transform is applied to each block. The subbands LH and HL are mixed via the type:

$$LHHL_n = \sqrt{LH_n^2 + HL_n^2}\ ,\ n = 1, 2, ..., N \tag{1}$$

producing the $LHHL$ subband, for each level of decomposition n. In our study, we used the ordinary number of $N = 3$ decomposition levels, which has prevailed as a kind of standard practice in the literature. The block's dimensions are relative to the size of the texture pattern embedded in a given image and can take typical values of 8×8 or 16×16.

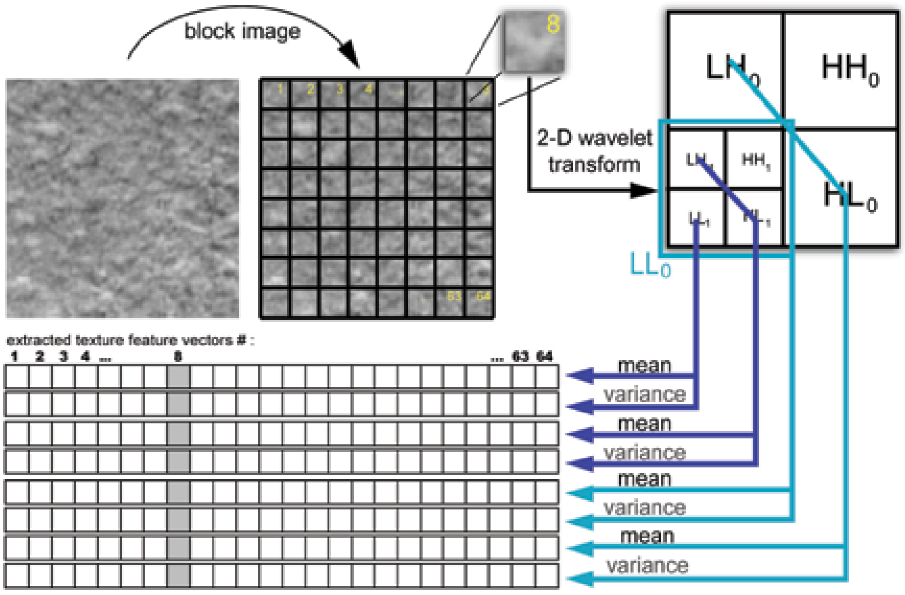


Fig. 1. Texture feature extraction using wavelet transform. Each block of the texture image is decomposed and the mean and variance values are calculated over the LL_n and $LHHL_n$ subbands.

For each square patch, the mean $\mu_{[k,m]}$ and the variance $\sigma_{[k,m]}$ of the energy distribution of the transform coefficients are calculated, as presented in (Pothos et al., 2007). Grouping together the mean and variance values of the same $m-$th block from all the subbands, a vector of $2\times(2\times N)$ coefficients is produced that describes the local texture information of the specific block. In general we will have a set of M such vectors that best describe the texture information of the entire image. A representative scheme that sketches the above procedure is illustrated in Fig. 1.

3.2 Discrete Cosine Transform (DCT)

The Discrete Cosine Transform (DCT) has been widely used in the literature for efficient texture feature selection. It uses cosines of varying spatial frequencies as basis functions and is commonly known for its implementation in the JPEG compression standard (Bhaskaran & Konstantinides, 1995). The DCT coefficients are obtained covering different spectral bands. For texture images, much of the signal energy lies at low-frequency components, which appear in the upper left corner of the DCT. Knowing that DCT converts the spatial information into the frequency domain, texture features can be defined as the spectrum energies in different localizations of a local block. Since the DC coefficient represents (almost) the average grayscale value of each $N\times N$ block, it is not considered to carry any texture information. The remaining AC coefficients capture the details - or frequency and directionality properties - within the pixel-block and therefore can be considered to characterize image texture and be utilized as texture features.

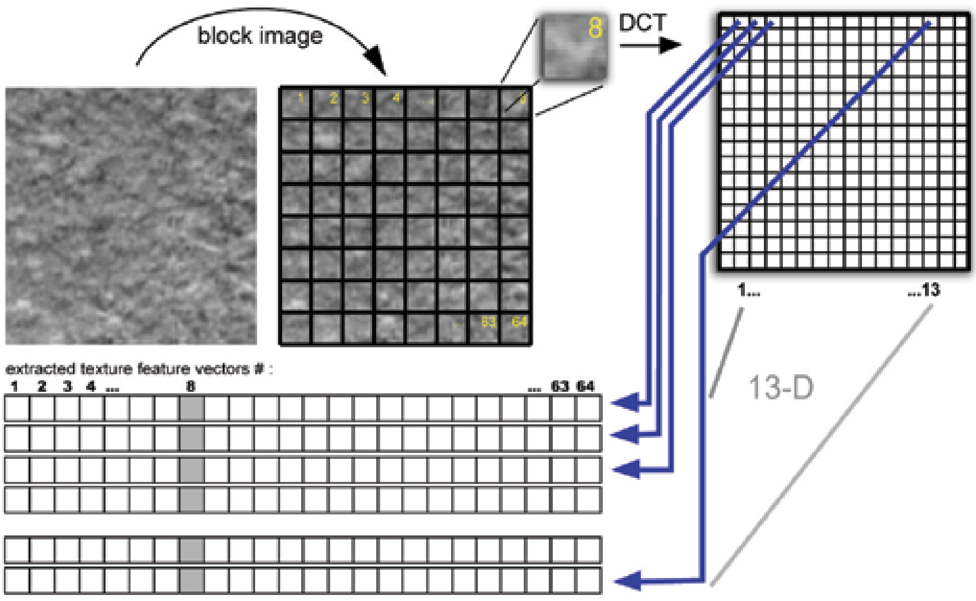


Fig. 2. Texture features extraction using DCT. DCT is applied on each block of the texture image and a feature vector is creating via summing up the square values of the coefficients in the diagonals of the block.

In order to extract textural attributes, the images are initially partitioned into $N\times N$ pixel-blocks, with $N=16$ in our case. The block size was selected in order to reduce the number of extracted feature vectors and also try to effectively capture the texture information using a larger image patch. In addition, it was experimentally verified to produce enhanced classification results compared to a smaller pixel-block (e.g., $N = 8$). Then, the DCT is applied to each distinct block, as illustrated in Fig. 2. From each DCT block, texture can be now represented by a feature vector V_m, with $m \in [1, 2N-2]$, the elements of which are the square sums of coefficients of the corresponding diagonals (i.e., zig-zag traversal lines). The vector resulting from the zig-zag ordering contain all the AC coefficients starting from the upper left location (i.e., $(0,1)$) to the bottom right (i.e., $(N-1, N-1)$). Assuming that a given image is initially divided into M blocks of 16×16 pixels, then a set of M feature vectors can be extracted that best describes the texture image content of the particular image. The specific indexing scheme was found to be robust, when similarity-based image rotation is considered (Theoharatos et al., 2006).

3.3 Gabor filters

The relation between the human vision system and the Gabor filters is a strong motive to test Gabor filtering for texture feature extraction. Spatially, a Gabor function is a Gaussian modulated sinusoid. In his work (Daugman, 1985), Daugman generalized the Gabor function to the model of the 2-D form:

$$G(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\pi\left[\frac{(x-x_0)^2}{\sigma_x{}^2} + \frac{(y-y_0)^2}{\sigma_y{}^2}\right]} e^{i(\xi_0 x + v_0 y)} \tag{2}$$

where (x_0, y_0) is the center of the field in the spatial domain and (ξ_0, v_0) is the optimal spatial frequency of the filter in frequency domain. σ_x and σ_y are the standard deviations of the elliptical Gaussian for axis x and axis y respectively.

In order to extract texture information, we first partitioned the images into M non-overlapping blocks. Then, the Gabor filters were applied using 4 scales and 6 orientations, creating $N = 24$ filtered subimages. These subimages are obtained by computing the magnitude from the real $G_{\Re_n}$ and imaginary $G_{\Im_n}$ parts of each n subbands:

$$G_n = \sqrt{G_{\Re_n}^2 + G_{\Im_n}^2}\ ,\ n = 1, 2, ..., N \tag{3}$$

Mean and variance are calculated by G_n for each one of the $N-$filtered subimages. In (Zhang et al., 2000), a $2 \times N - D$ multidimensional vector is constructed such that to be used for similarity matching using a valid (dis)similarity measure (i.e., the sum of Euclidean distances). In this study, a 2×24 dimensional feature vector is built for the description of the texture information, corresponding to the mean and variance values per filtered subimages that are contained in each corresponding block. In the final stage, a total number of M feature vectors of $48 - D$ is constructed for the description of the texture information of all database images. A representative scheme of the previously reported technique is illustrated in Fig. 3, clearly sketching the overall procedure.

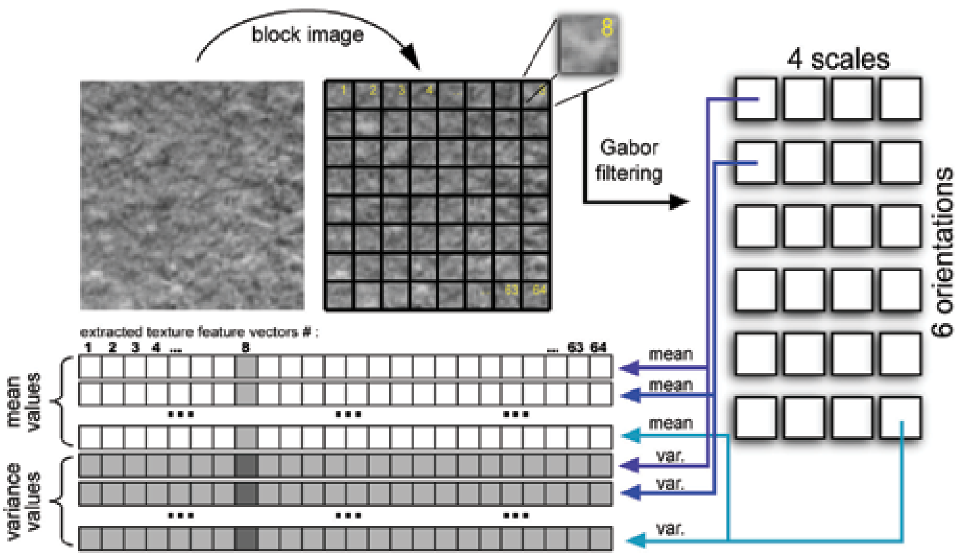


Fig. 3. Texture features extraction using Gabor filters. Each image is blocked into M blocks, and from each block 48 values are extracted, after applying 24 Gabor filters.

3.4 Edge Histogram descriptor

The Edge Histogram is a useful texture descriptor that captures the spatial distribution of edges in the image and is defined in MPEG-7 for similarity search and retrieval practices (Manjunath et al., 2002). The local distribution of edges is considered a reliable candidate attribute and useful for image to image matching, even when the underlying texture structure is not homogeneous in texture properties. The extraction of textural features is estimated as follows. In the first step, the image is divided into 4×4 subimages and the local-edge distribution for each subimage is represented by a histogram. To do this, edges in the subimages are categorized into five types: vertical, horizontal, $45°$ diagonal, $135°$ diagonal and nondirectional (or isotropic). Therefore, each image is comprised of 16 subimages with five bins each (corresponding to the above five category-types). The overall histogram is composed of $16 \times 5 = 80$ bins.

Edge detection and classification in each subimage can be done by further dividing those subimages into non-overlapping square blocks (e.g., into a number of 2×2 pixel-images) and afterwards apply appropriate oriented edge detectors (including four directional selective detectors and one isotropic operator (Manjunath et al., 2001)) to compute the corresponding edge strengths. If the maximum edge strength of these oriented edge detectors is found above a given threshold, then the corresponding edge orientation is associated with the image-block which is considered to be an edge-block. If not, then the image-block is not classified as edge-block. Finally, the edge blocks that result contribute to the appropriate binning procedure of the histogram descriptor, with each bin value normalized to the total number of image-blocks in the subimage (i.e., $[0,1]$). The individual algorithmic steps of the local-edge histogram descriptor are summarized in Fig. 4, with $h(i)$ denoting the overall histogram comprised of 80 bins.

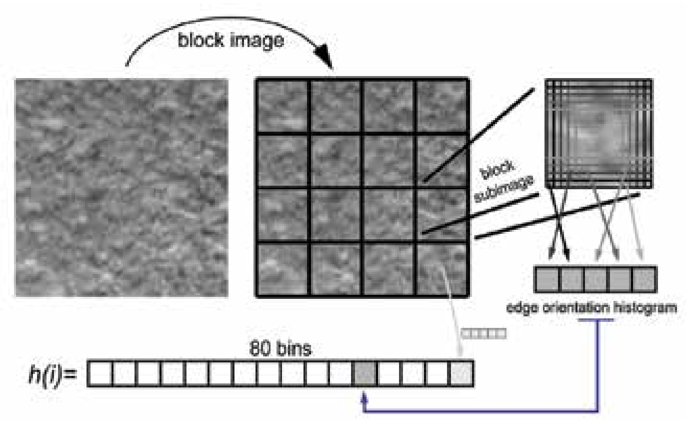


Fig. 4. Construction of local-edge histogram $h(i)$.

However, the local-edge histogram is not sufficient enough for effective image matching. For this reason, global-edge distributions are used in association to local-based ones (Manjunath et al., 2002). In this way, global- and semiglobal-edge histograms are produced respectively, computed directly from the 80 local histogram bins. Regarding the global-edge histogram, the five types of edge distributions in all subimages are accumulated. For semiglobal-edge histograms subsets of subimages are grouped as shown in Fig. 5. The combination of all those distinct histograms, produce a histogram of 150 bins. For matching the extracted features, MPEG-7 defines the $L_1-\text{norm}$ $D(A,B)$ as the distance measure for comparing two image histograms A and B, using the following formula:

$$D(A,B)=\sum_{i=0}^{79}|h_A(i)-h_B(i)|+5\times\sum_{i=0}^{4}|h_A^g(i)-h_B^g(i)|+\sum_{i=0}^{64}|h_A^S(i)+h_B^S(i)| \tag{4}$$

where $h_A(i)$ and $h_B(i)$ represent the normalized edge histogram bin values of image *A* and image *B* respectively. In the above equation, $h_A^g(i)$ and $h_B^g(i)$, as well as $h_A^S(i)$ and $h_B^S(i)$ represent the normalized bin values for the global-edge and semiglobal-edge histograms respectively, of consecutive images *A* and *B*.

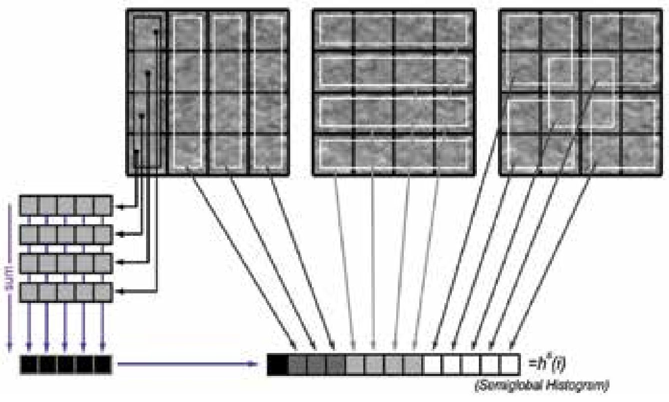


Fig. 5. Construction of semi-global edge histogram $h^S(i)$.

In order to measure the similarity of the different texture images using distributional (dis)similarity measures, the extracted edge histogram data need to be re-arranged into a form of multiple vectors, instead of a histogram. We perform the multivariate nature our distributional measures, by operating in five dimensions as follows. Firstly, we calculate separately each edge histogram (see Fig. 4) from each subimage of the local-edge histogram, resulting in 16 vectors of five dimensions (i.e., $5-D$) each. Additionally, an extra one $5-D$ vector is computed from the global-edge histogram. Finally, we utilize the 13 parts of the semiglobal-edge histogram, in order to create another 13 vectors of $5-D$ each. The ensemble of these practises produce a total of 30 vectors in $5-D$ and thus the distributional (dis)similarity measures can now straightforwardly applied for texture matching over the selected attributes of two separate images. We note here that, each dimension of the formed feature space corresponds to the normalized number of edges that are found to be vertical, horizontal, diagonal, diagonal and nondirectional.

4. Distance between distributions

In order to compare the extracted spectral representations in pairs and therefore estimate the similarity between texture images, we resorted to the field of multivariate statistics for choosing proper distributional-based measures. There exists a rich literature on probability distribution distance measures (Do & Vetterli, 2002; Theoharatos et al., 2006; Rubner et al., 2001; Gibbs & Su, 2002), the choice of which is relatively influenced by a number of inherent parameters of the data distribution. These include the amount of existing data, the dimensionality of the resulted feature space and the distributional structure of the selected multidimensional vectors. In applications like texture classification, face recognition, fingerprint identification etc., intrinsic representations that constitute characteristic manifolds coming from the corresponding high-dimensional data distributions are realized. The adaptation of valid/proper (dis)similarity measures capable for capturing these structures and, moreover, performing pairwise comparisons between pairs of suchlike distributions is the key factor for successfully assessing huge data archives. Several well known distance measures were examined towards achieving the above goal. In our analysis that follows, we present results for the best (and simultaneously more appropriate) two ones, i.e. the multivariate Wald-Wolfowitz test (WW-test) (Friedman & Rafsky, 1979) and the Kantorovich-Wasserstein distance (Gibbs & Su, 2002). In their basic implementation, both measures utilize the L_2-norm for calculating the ground distance between sample vectors. For comparison reasons, histogram-based measures are also utilized in several experimental trials such as Histogram Intersection (HI), Kullback-Leibler Divergence (KL-D) and Jeffrey Divergence (J-D).

4.1 The multivariate Wald-Wolfowitz test (WW-test)

As an important constituent of the introduced framework, a nonparametric test is adopted for estimating texture content similarity in a reliable and convenient way. It is a multivariate extension of the classical Wald-Wolfowitz test, comparing two different sets of points in $\Re^p$ by checking whether they form different branches in the overall MST-graph (Zahn, 1971). The output can be expressed as the probability that the two samples are coming from the same distribution (Friedman & Rafsky, 1979).

In the multivariate case, the graph is built over points in $\Re^p$: a single node corresponds to every given point and the weight associated with every possible edge is the corresponding interpoint norm (i.e., ground distance used in its basic implementation). The edges involved in the construction of MST are the ensemble of straight-line segments connecting all points with minimum total length. WW-test can be used to test the hypothesis $\mathbf{H}_o$, whether any two given multidimensional point samples $\{X_i\}_{i=1:m}$ and $\{Y_i\}_{i=1:n}$ come from the same multivariate distribution. At first, the two data samples of size m and n are considered, respectively, from distributions defined in $\Re^p$. Then, the sample identity of each point is not encountered and the MST of the overall sample is constructed. Based on the sample identities of the points, a test statistic R is computed that is defined as the number of disjoint subtrees that finally result. Rejection of $\mathbf{H_o}$ is for small values of R. It has been shown that the quantity:

$$W = \frac{R - E[R]}{\sqrt{Var[R \mid C]}} \tag{5}$$

approaches the standard normal distribution, while the mean $E[R]$ and variance $Var[R \mid C]$ of R are given in closed form (Friedman & Rafsky, 1979). Its importance is that using simple formulae, the significance level (and p-value) for the acceptance of the hypothesis $\mathbf{H_o}$ can be readily estimated. WW-test, based on the MST-planning procedure, offers a unique environment for contrasting different signal representations. Thus, it can effectively cope with the understanding and matching of manifold-type structures, which is actually the case of the vectorial spectral representations of the texture images under study.

4.2 Kantorovich-Wasserstein distance

The Kantorovich-Wasserstein (KWass) metric defines a "distance" between two stochastic distributions. It is described by the formula:

$$d_w(\mu,\nu) = \inf_{J}\{\mathbf{E}[d(X,Y)] : \mathrm{L}(X) = \mu, \mathrm{L}(Y) = \nu\} \tag{6}$$

where the infimum is taken over all joint distributions J with marginals μ and ν (Gibbs & Su, 2002).

For discrete distributions X, Y with samples of the same size $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$, the minimum is taken over all permutations. It is common to use the Hungarian algorithm in order to solve the optimal assignment problem (Levina & Bickel, 2001).

4.3 Distances between histograms

In order to provide a short description of the other methods employed here for comparison purposes, let $H = \{h_i\}$ and $K = \{k_i\}$ be histograms from two texture images H and K to be compared respectively, each containing n bins. A plethora of measures have been reported in the literature for calculating the distance between histogram distributions (Rubner et al., 2001). Here, three characteristic measures are utilized that are commonly used by other researchers.

Histogram Intersection (HI)
It was originally proposed for color image retrieval (Swain & Ballard, 1991) in the spatial domain and is found to be attractive due to its ability to handle partial matches (Rubner & Tomasi, 2001). The HI-measure is given by:

$$d_{HI}(H,K) = 1 - \frac{\sum_{i=1}^{n} \min(h_i, k_i)}{\sum_{i=1}^{n} k_i} \tag{7}$$

Kullback-Leibler Divergence (KL-D)
It measures how inefficient on average it would be to code one histogram using the other as the code-book (Rubner et al., 2001):

$$d_{KL}(H,K) = \sum_{i=1}^{n} h_i \log \frac{h_i}{k_i} \tag{8}$$

Jeffrey Divergence (J-D)
This is a modification of the KL-D that is symmetric, numerical stable and robust with respect to noise and size of histogram bins (Rubner & Tomasi, 2001), given by:

$$d_{JD}(H,K) = \sum_{i=1}^{n} \left(h_i \log \frac{h_i}{m_i} + k_i \log \frac{k_i}{m_i} \right), \tag{9}$$

where $m_i = h_i + k_i / 2$.

5. Experimental analysis

For our experimental analysis two texture databases were utilized: the OUTex (University of Oulu Texture database) and the Photometric texture database. The first dataset contains 24 distinct texture categories, having 180 grayscale images of similar size in each class and thus resulting in a total number of 4320 texture images. The amount of images comprising a single category is formed by using nine different texture orientation images. From those 4320 images, 216 were selected as queries (i.e., one texture for each orientation and category), while the leave-one-out procedure was used in a $k-$NN for the classification procedure. The incorporation of a bigger number of query-images had a very slight impact on the classification results. The second dataset contains 34 different kinds of textures, having 56 images each. For this database, we used an increasing number of images per texture in the database (to train the classifier), while the rest images were utilized as query-images to be classified. Fig. 6 presents characteristic samples from all texture categories for both the OUTex and Photometric texture databases.
In Fig. 7, the classification results are given for the OUTex database using the leave-one-out procedure. The horizontal axis contains the values of k of the simple $k-$NN classifier, while the vertical one contains the classification error rate of each method. In all cases, the (distributional-based) multivariate WW-test outperforms significantly all other (histogram-based) measures. The DCT and Gabor feature extraction methods seem to produce the best classification results, while the edge histogram one seems to provide poor outcomes.

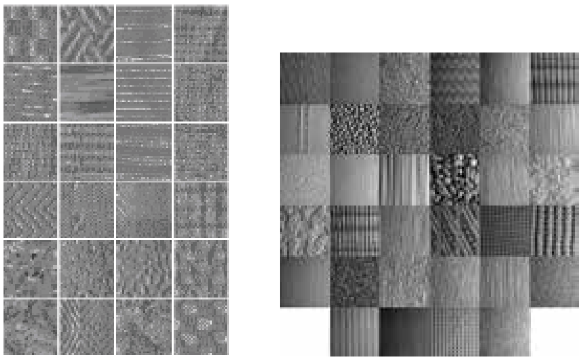


Fig. 6. Sample category images of the utilized texture databases. Left) OUTex database. Right) Photometric texture database.

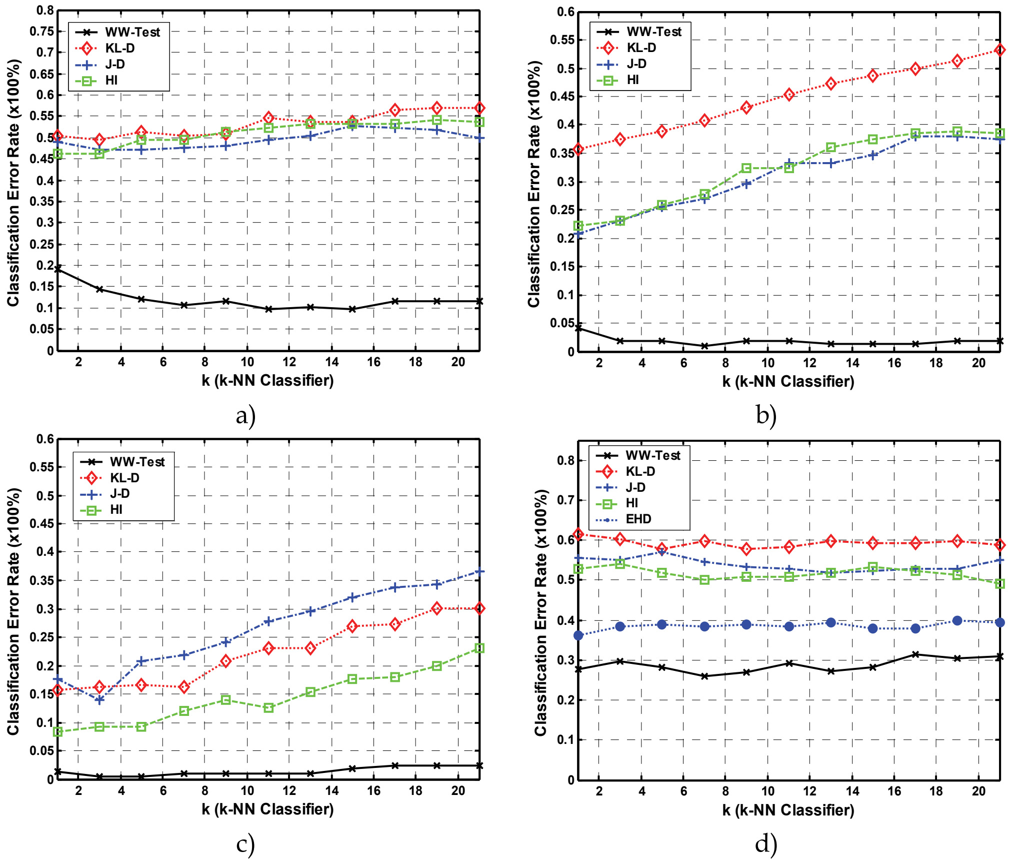
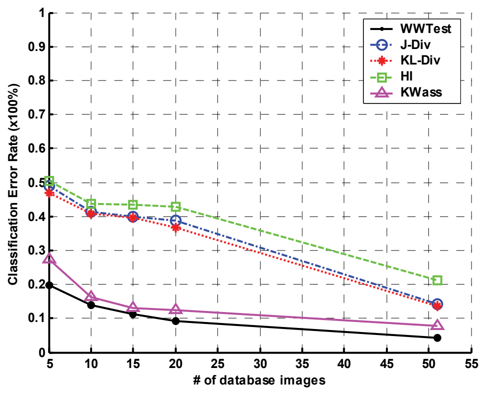


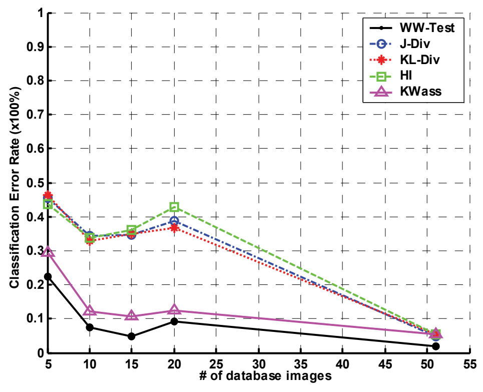
Fig. 7. Classification Error Rate results for a) Wavelets, b) DCT, c) Gabor filters and d) Edge Histogram, for the OUTex database.

The OUTex Database downloaded from: http://www.outex.oulu.fi/temp/
The Photometric Database downloaded from: http://www.taurusstudio.net/research/pmtexdb/index.htm

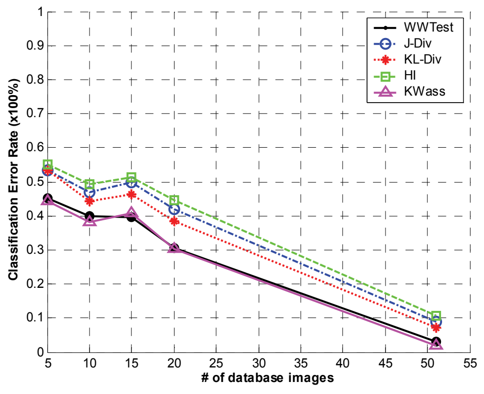
In Fig. 8 the classification results over the Photometric texture database are exemplified. The horizontal axis contains the number of texture images used as database images for the $k-$NN classifier. In this way, it is possible to understand how the numbers of images used for "training" the classifier affects the results. In practice, the most important outcome would be in the case of using only 5 training images, which is more close to real application implementations. Wavelets and DCT seems to give best results for a few training images, while Gabor filters and edge histogram seems to have high classification error ratio. The use of many images for training the classifier affects the results, producing low classification error rate for all methods, except for the edge histogram that is in accordance with the results produced using the OUTex database. We have to notice here that, in general, the multivariate WW-test performs slightly better than the KWass measure, when distributional-based measures are considered. In addition, histogram-based measures do not perform adequately when compared to distributional-based ones, in all different texture feature extraction methodologies. This internal comparison is in accordance with the results coming from both texture datasets, as also theoretically expected from our initial analysis and the reports coming from the wide literature.



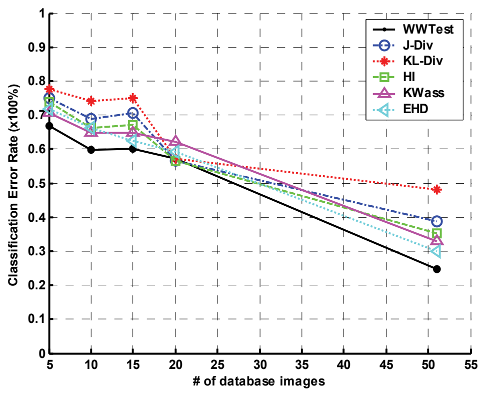
a)



b)



c)



d)

Fig. 8. Classification Error Rate results for a) Wavelets, b) DCT, c) Gabor filters and d) Edge Histogram for different number of training images, for the Photometric database.

6. Boosting the classification performance via Vector Quantization

Vector quantization aims at representing the data with a reduced set of prototype data vectors and thus summarizes the input information while inducing minimal distortion. In the case of texture images, a codebook of k code vectors that summarizes the vectorial representation of the entire spectral information is designed by applying the Neural-Gas algorithm to the data matrix. This algorithm is an artificial neural network model, which converges efficiently to a small, user-defined number of codebook vectors, using a stochastic gradient descent procedure with a "soft-max" adaptation rule that minimizes the average distortion error (Martinez et al., 1993).

The Neural Gas network operates by utilizing first a stochastic sequence of incoming data vectors $X(t)$, which is governed by the distribution $P(\mathbf{X})$ over the manifold $\mathbf{V}$. Then, an adaptation step is performed for adjusting the weights of the k neurons $\{A_j\}_{j=1:k}$:

$$\Delta A_j = \varepsilon \cdot h_\lambda\left(f_j\left(X(t),\{A_i\}_{i=1:k}\right)\right)\left(X(t)-A_j\right),\ j=1,2,\ldots,k,\ \forall\, t=1,\ldots,t_{\max} \tag{10}$$

The function $h_\lambda(y)$ in the above equation has an exponential form $e^{-y/\lambda}$ and $f\left(X,\{A\}_j\right)$ is an indicator function that determines the 'neighborhood-ranking' of the reference vectors according to their distance from the input vector $\mathbf{X}$, while for both parameters ε and λ an exponential decreasing schedule is followed, with $t_{\max}$ being the final number of adaptation steps that can be defined from the data based on simple convergence criteria.

The asymptotic density distribution of the codebook vectors $P(\mathbf{A})$ was proved, mathematically (Martinez et al., 1993), to be proportional to the data density distribution $P(\mathbf{A}) \propto P(\mathbf{X})^{\underline{d}/(\underline{d}+2)}$ where $\underline{d} \leq d$ is the intrinsic dimensionality of the input data. This theoretical evidence along with the accompanying experimental evidence (Martinez & Schulten, 1994) motivated our conjecture that the designed codebook could serve as a faithful representation of the vectorial distribution in color-space and therefore could be utilized in the subsequent comparisons regarding color content. The relationships between filter responses are encoded in the joint multivariate distribution and provide unique information about the textural structure. To reduce the complexity of the classification problem, texture distribution comparisons are carried out in pairwise fashion using the distribution-distance measures presented before to test the efficiency of the sequential algorithmic procedure.

In order to evaluate the classification accuracy of our procedure when incorporating the Neural-Gas based vector quantization scheme, the Photometric database was utilized. In our experiments, 4 training sets of images were used for the classifier, with each one containing 5, 20, 35 and 51 database images respectively per class, while the rest ones were used as queries. Experiments took place for Wavelets, DCT and Gabor methods, which were proven to produce the best classification results. Each image was first partitioned into several overlapping blocks. For each block, all three feature extraction methods were applied, as previously explained. Due to the block-overlapping nature of our procedure, the extracted texture signatures are comprised of a large number of feature vectors. The Neural-Gas algorithm is next used to select the most representative ones that best describe the texture feature distribution, boosting in this way the classification results. In Fig. 9 the general scheme of using Neural-Gas is presented.

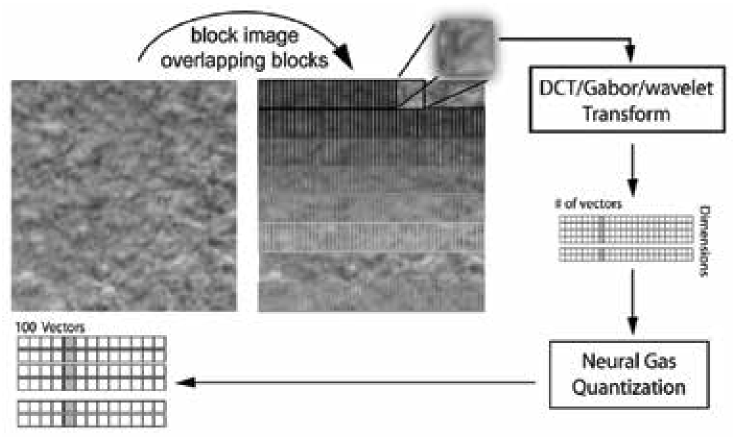


Fig. 9. Overlapping blocks produce a large number of feature vectors. Neural-Gas is used to select the most representative ones that best describe the image texture.

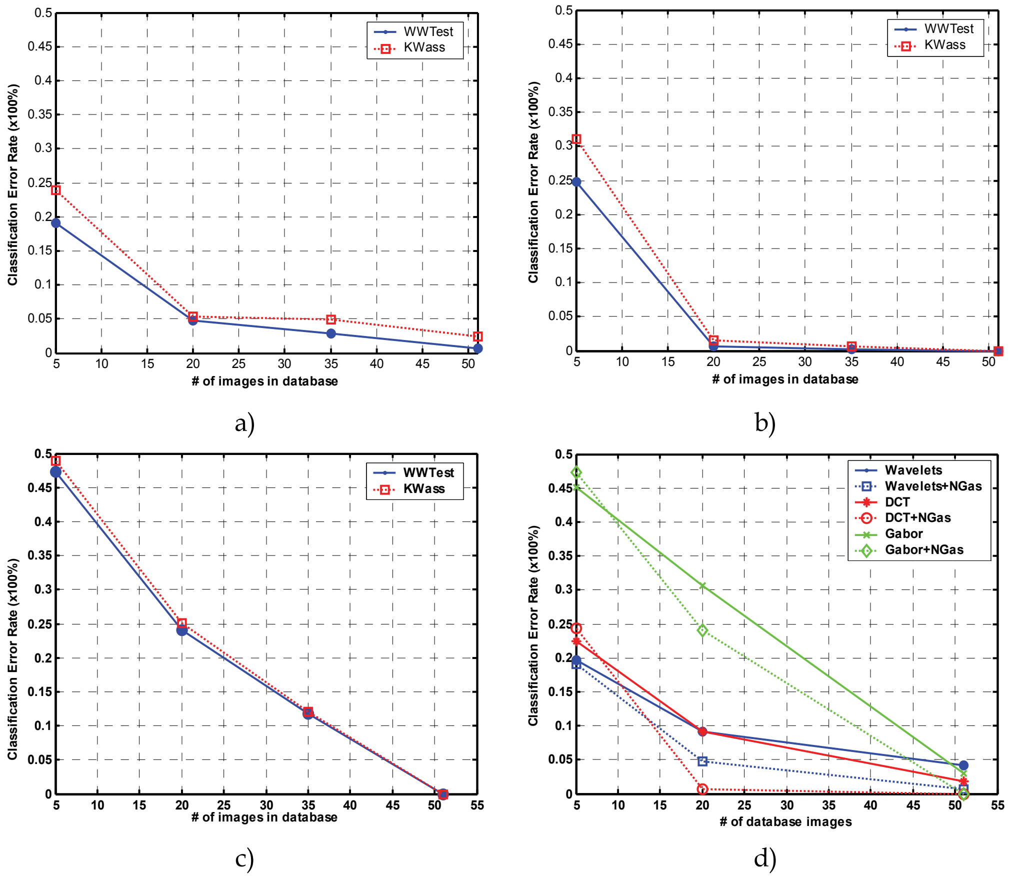


Fig. 10. Experimental results using the Neural-Gas algorithm for a) Wavelets, b) DCT, c) Gabor filters. d) Comparative results of the methods with and without Neural-Gas.

In Fig. 10, the results obtained utilizing the Neural-Gas are presented. When the classifier is trained with a few texture images, the Neural-Gas procedure seems to slightly increase the classification error. In contrast, as the number of database samples is increased, the Neural-Gas procedure boosts the accuracy of the classification quite enough.

7. Conclusion

A robust methodology is presented in this chapter that tries to tackle the texture classification problem. The multivariate WW-test and the KWass distance are used in order to measure how close two distributions are. Their generic character stems from the fact that, by altering the character of texture image characteristics, we can modify the flavour of formulated queries. To avoid problems associated with histograms as empirical estimates of the distribution (e.g., the binning effect), particularly in high-dimensional spaces, dissimilarity between texture distributions is computed using distributional-based multivariate analysis. These multidimensional measures can adequately operate even with a small number of distributional samples and is well suited for texture matching.

Individual texture samples were extracted from the images of the OUTex and the Photometric texture databases, by partitioning the image into regions of almost homogenous texture content. The intrinsic dimensionality of the texture regions is computed by means of image decomposition implementing some of the well-established techniques. Depending on the technique, the joint spectral distribution is sampled in the form of multivariate vectors. The efficiency in textural feature extraction of the different methods, as well as the competence of the above measures in distributional texture image representations, was tested with quite satisfactory results, which yield future ideas for research and application. In addition, a neural-network based vector quantizer was adopted in order to further boost the classification accuracy of the introduced methodologies. Each texture image distribution was summarized by a vector quantization scheme in order to select a restricted number of prototype code vectors, thus resulting in a sampled representation of the original spectral distribution. These code vectors played the role of a spectral signature for each texture image, capturing its basic structure and providing a sparse-compact representation.

As a scheduled extension of our work, the application of the introduced methodology to color or multispectral images for texture classification can be straightforwardly implemented, by placing other subband's texture information to higher dimensions in the feature space. However, special care has to be taken to the number of extracted feature vectors in the case of multispectral images, due to the "curse of dimensionality" (Costa & Hero, 2004). In addition, more has to be done in order to overcome the problem that arises from the weakness to capture texture patterns that have different plain scales. This might be accomplished by utilizing adaptive scalable blocks, by means of – each time – different sized block patches inside the input images. In this way, the texture information can be efficiently captured in different scales, making use of possible regions of interest. Moreover, texture segmentation algorithms could be incorporated towards this solution, among the plethora that is available in the literature the recent years (Jain & Farrokhnia, 1991; Manjunath & Chellappa, 1991). Finally, the use of alternative ways to extract texture information based on high-level features (i.e., semantic-based texture attributes), so as to correlate better with the human perceptual inspection, is of crucial importance (Liu et al., 2007), as well as the potential application of combining other primitives in the feature extraction methodology, such as color and shape.

8. Acknowledgements

This work was financed by the European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II), and particularly the Program "New graduate programs of University of Patras".

9. References

Bhaskaran, V. & Konstantinides, K. (1995). *Image and video compression standards, algorithms and architectures,* Kluwer Academic Publishers, ISBN: 0-7923-9952-8, Norwell, Massachusetts USA

Costa, J. & Hero, A.O. (2004). Geodesic entropic graphs for dimension and entropy estimation in manifold learning. *IEEE Trans. on Signal Processing,* Vol. 52, No. 8, pp. 2210-2221

Daugman, J.G. (1985). Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Journal of the Optical Society of America A,* Vol. 2, No. 7, pp. 1160-1169

Do, M.N. & Vetterli, M. (2002). Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance. *IEEE Trans. on Image Processing,* Vol. 11, No. 2, pp. 146-158

Friedman, J.H. & Rafsky, L.C. (1979). Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests. *Annals of Statistics,* Vol. 7, No. 4, pp. 697-717

Gibbs, A. & Su, F. (2002). On choosing and bounding probability metrics. *International Statistical Review,* Vol. 70, No. 3, pp. 419-435

Jain, A.K. & Farrokhnia, F. (1991). Unsupervised texture segmentation using Gabor filters. *Pattern Recognition,* Vol. 24, No. 12, pp. 1167–1186

Laine, A. & Fan, J. (1993). Texture classification by wavelet packet signatures. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 15, No. 11, pp. 1186–1190

Leow, W.K. & Li, R. (2004). The analysis and applications of adaptive-binning color histograms. *Computer Vision and Image Understanding,* Vol. 94, No. 1-3, pp. 67–91

Leung, T. & Malik, J. (2001). Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision,* Vol. 43, No. 1, pp. 29-44

Levina, E. & Bickel, P. (2001). The earth mover's distance is the Mallows distance: some insights from statistics. *Proceedings of IEEE Int. Conf. on Computer Vision,* pp. 251-256, Vancouver, Canada, July 2001

Liu, Y.; Zhang, D.; Lu, G. & Ma, W.Y. (2007). A survey of content-based image retrieval with high-level semantics. *Pattern Recognition,* Vol. 40, No. 1, pp. 262-282

Manjunath, B.S. & Chellappa, R. (1991). Unsupervised texture segmentation using Markov random fields models. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 13, No. 5, pp. 478–482

Manjunath, B.S.; Ohm, J.R.; Vasudevan, V.V. & Yamada, A. (2001). Color and texture descriptors. *IEEE Trans. on Circuits and Systems for Video Technology,* Vol. 11, No. 6, pp. 703-715

Manjunath, B.S.; Salembier, P. & Sikora, T. (2002). *Introduction to MPEG-7: multimedia content description interface,* Wiley, ISBN: 978-0-471-48678-7, New York

Martinez, T.M. & Schulten, K.J. (1994). Topology representing networks. *Neural Networks,* Vol. 7, No. 3, pp. 507-522

Martinez, T.M.; Berkovich, S.G. & Schulten, K.J. (1993). "Neural-gas" network for vector quantization and its application to time-series prediction. *IEEE Trans. on Neural Networks,* Vol. 4, No. 4, pp. 558-569.

Ojala, T.; Pietikainen, M. & Harwood, D. (1996). A comparative study of texture measures with classification based feature distributions. *Pattern Recognition,* Vol. 29, No. 1, pp. 51–59

Pothos, V.K.; Theoharatos, C.; Zygouris, E. & Economou, G. (2007). Distributional-based texture classification using non-parametric statistics. *Pattern Analysis and Applications, Special Issue on Non-parametric Distance-based Classification Techniques and its Applications,* DOI 10.1007/s10044-007-0083-9

Randen, T. & Husoy, J.H. (1999). Filtering for texture classification: a comparative study. *IEEE Trans on Pattern Analysis and Machine Intelligence,* Vol. 21, No. 4, pp. 291-310

Rubner, Y. & Tomasi, C. (2001). *Perceptual metrics for image database navigation,* Kluwer Academic Publishers, ISBN: 0-7923-7219-0, Norwell, Massachusetts, USA

Rubner, Y.; Puzicha, J.; Tomasi, C. & Buhmann, J.M. (2001). Empirical evaluation of dissimilarity measures for color and texture. *Computer Vision and Image Understanding,* Vol. 84, No. 1, pp. 25–43

Sebe, N. & Lew, M.S. (2000). Wavelet based texture classification. *Proceedings of IEEE Int. Conference on Pattern Recognition,* pp. 947-950, Barcelona, Spain, September 2000, IEEE Computer Society Washington, DC

Swain, M.J. & Ballard, D.H. (1991). Color indexing. *International Journal of Computer Vision,* Vol. 7, No. 1, pp. 11-32

Theoharatos, C.; Pothos, V.K.; Laskaris, N.A.; Economou, G. & Fotopoulos, S. (2006). Multivariate image similarity in the compressed domain using statistical graph matching. *Pattern Recognition,* Vol. 39, No. 10, pp. 1892-1904

Zhang, D.S.; Wong, A.W.; Indrawan, M. & Lu, G. (2000). Content-based image retrieval using Gabor texture features, *Proceedings of IEEE Pacifin-Rim Int. Conference on Multimedia,* pp. 392-395, University of Sydney, Australia, December 2000

Recent Developments in Bit-Parallel Algorithms

Pablo San Segundo, Diego Rodríguez-Losada and Claudio Rossi
Universidad Politécnica de Madrid
Spain

1. Introduction

A bit array (or bit vector, bitboard, bitmap etc. depending on its application) is a data structure which stores individual bits in a compact form and is effective at exploiting bit-level parallelism in hardware to perform operations quickly as well as reducing memory requirements. Working at bit level is nothing new: i.e. STL[1] for C++ has a *bitset* container as data type, and mapping pixels to bits or processes to priority queues in some operative systems are but two examples of an interminable list of applications where space requirements are critical.
However, to improve overall efficiency by bit-masking operations is *hard* in any scenario. One obvious reason for this is that bit vectors are compact data structures difficult to manipulate, all the more so since extracting information relative to a single bit of the array has an overhead which does not exist in a classical implementation. From a theoretical perspective there have been some important complexity results concerning bit-paralellism, where modern CPUs are seen as non deterministic Touring Machines with power limited to the size of its registers (denoted as w_{size}). In practice, bit-parallelism has become an important tool for domains such as string matching as in (Baeza-Yates R. and Gonnet G. H 1992), where the complexity of a linear algorithm is reduced by a factor w_{size}. It is important to note that these successes have not extended to more complex NP combinatorial problems in the general case, a key issue and a topic which has been a line of research of the authors in recent years.
A classical search domain for bit vectors has been board games, the origin of the term *bitboard*. In chess-playing programs, the bits in a 64-bit bitboard map to a particular Boolean property concerning the 64 squares of the chessboard (cf. Heinz E.A. 1997). One of the first systems to employ bitboards as the basic modelling unit is the KAISSA chess programming team in the Soviet Union during the late 1960s. Today almost all relevant chess programs employ this form of encoding and reason, at least partly, over a bit vector space.
This chapter covers the use of bit-parallelism as an AI tool to implement efficient search procedures. It focuses on fully bit encoded search domains, where declarative frame knowledge is mapped to bit vectors and procedural frame knowledge (i.e. basic transition operators etc.) is mapped to simple bitwise operations over a bit vector search space.

[1] STL: Standard Template Libraries

The material presented is structured in three parts. Section 2 covers exact (optimal) search. It focuses on a depth-first search algorithm to show the advantages and disadvantages of search in a bit vector space w.r.t. to a classical encoding, including experiments. Based on these experiments and recent work of the authors on the maximum clique problem (San Segundo P. et al. 2007) the section highlights the strength of simple graph models as a tool for implementing efficient bit parallel search procedures in general, and NP-hard problems in particular. At the end of the section the Boolean satisfiability problem and the N-Queens problem are suggested as new candidates for bit-parallel search.
Section 3 covers bit-parallel search in non-exact scenarios. In particular an efficient genetic algorithm for SAT is compared with an equivalent bit parallel version. The section also includes computational experiments. Section 4 describes two real life applications where bit-parallelism has been applied with success, taken from the vision and robotics domain. Conclusions as well as a brief discussion of future work are stated in Section 5.

2. Exact search in a bit vector space

This section covers exact (optimal) bit-parallel efficient search procedures. It is assumed that the search domain can be fully bit encoded and that a reasonable bit encoding has already been found. The subject of how to find one such bit representation for a particular domain is out of the scope of this Section (and of the Chapter itself). Rather, the Section focuses on implementation and complexity issues related to systematic bit-parallel search. As case study the maximum clique problem has been selected for a number of reasons that will be explained throughout the section.

2.1 Basic bit operators

A typical fully bit encoded search space maps bits to domain entities and states to a number of bit vectors which represent Boolean properties of these entities. Without loss of generality, it can be assumed that non Boolean properties which describe a particular state can be reduced to a collection of Boolean ones. In this scenario, a bit vector is a {0,1} collection of cardinality the number of domain entities. A possible declaration of this data structure in C language can be found in figure 1.

```
typedef unsigned long long  BITARRAY;

/*bit vector declaration*/
BITARRAY bitvector [Cardinality];
```

Figure 1. Declaration of a bit vector in C language

Since bit vectors map to sets, bitwise operations are needed to compute the fundamental operators related to set theory. Table 1 shows basic bitwise operations for sets using C style syntax (i.e., &, |, ^ and ~ map onto AND, OR, XOR and NOT respectively). Note that the last operator in Table 1 is not an assignment over sets A and B, but a truth assertion.
A fully encoded bit-parallel algorithm employs a bit vector (possibly more than one) to guide search in the bit space. In any systematic bit-parallel search procedure two classical

bitwise operators stand out over the rest: A) operator *LSB*[2] (alias Bit Scan Forward or simply Bit Scan) which finds the first 1-bit in a bit vector, and B) operator *PC* (*P*opulation Count) which returns the number of 1-bits in a given bit vector. The former (or its counterpart MSB[3]) is typically used in node selection strategies whilst the latter is necessary for leaf node detection (typically the empty bitboard, $PC(BB)=0$). Notation throughout this paper includes an additional subindex to LSB or BB to make cardinality explicit (e.g. LSB_{64} refers to a bit scan over a 64 bit array).

$A \cap B \equiv A_{BB}\ \&\ B_{BB}$	$A-(A\cap B) \equiv (A_{BB} \wedge B_{BB})\ \&\ A_{BB}$
$\overline{A} \equiv \sim A_{BB}$	$B-(A\cap B) \equiv (A_{BB} \wedge B_{BB})\ \&\ B_{BB}$
$A \cup B \equiv A_{BB} \mid B_{BB}$	$A-B \equiv A_{BB}\ \&\ (\sim B_{BB})$
$(A\cup B)-(A\cap B) \equiv A_{BB} \wedge B_{BB}$	$A \supseteq B \Leftrightarrow B_{BB}\ \&\ (\sim A_{BB}) == \phi$

Table 1. Correspondence from set theory operators to bitwise operations written in C language.

Depending on the processor HW architecture and compiler used, both operators might be available as built-in functions or *intrinsics*, but their use is always restricted to the size of the CPU registers (w_{size}). The extension to bit vectors of cardinality higher than w_{size} is conceptually trivial but needs to be done carefully because the impact in overall efficiency is high. SW implementations of w_{size} LSB and PC are needed when they are not available as *intrinsics* and there are a large number of solutions available in literature (cf. Warren H.S. Jr 2002). For PC we recommend precomputation of a lookup table for all 16 bit possible combinations. For LSB a nice hashing solution for a 64 bit register CPU can be found in (1). MN is one magic number from a De Bruijn sequence such as 0x07EDD5E59A4E28C2. Computation BB&(-BB) isolates a single 1-bit and the *, >> operations constitute a perfect hash function for the isolani to a 6 bit index. For a more detailed explanation we refer the reader to (Leiserson, C. t al. 1998).

$$LSB_{64}(BB) = [(BB\ \&\ (-BB))\cdot MN] >> 58 \tag{1}$$

A common assumption in bit encoded exact search models is that the benefits of parallelism at bit level have a counterpart in the overhead needed to extract information relative to a single bit in the compact bit array. This is, in fact, quite true in a general sense and is probably the reason why bit-parallelism has not attracted so much attention in AI real life applications as yet. This key issue is covered in the following subsection.

2.2 Complexity of bit scanning in bit-parallel systematic search

Finding a 1-bit in a compact bit array is an important overhead to be taken into account for efficient bit encoded exact search models. Worst case complexity for a naïve LSB_n computation is $O(n)$. A more efficient 16 bit direct lookup table implementation computes

[2] *LSB* stands for *L*east *S*ignificant *B*it

[3] *MSB* stands for *M*ost *S*ignificant *B*it

LSB_n in $O(\frac{4n}{w_{size}})$. For non bit-encoded models (e.g. an array indexed by the position of the element) the cost of a single LSB operation is in $O(n)$, clearly worse w.r.t. the bit model. However, the situation changes when the problem is extended to finding the first k-bits in a bit set (alternatively the first k elements in a list). In this case, worst case complexity for lists is still in O(n) whereas, although it is possible to index the w_{size} blocks of bits, there is no getting over the $LSB_{w_{sixe}}$ complexity of finding a 1-bit in a particular block. Worst case computation, assuming $k <= w_{size}$ and a 16-bit direct lookup table implementation of $LSB_{w_{size}}$ is:

$$O(Bit\,Scan\ \ for\,k\,bits) = \frac{4(N-1)k}{W_{size}} \tag{2}$$

which grows linearly with the number of bits to find. Figure 2 illustrates this inherent complexity showing time results for finding the first 100 1-bits in a random generated population of size 5000 with varying densities.

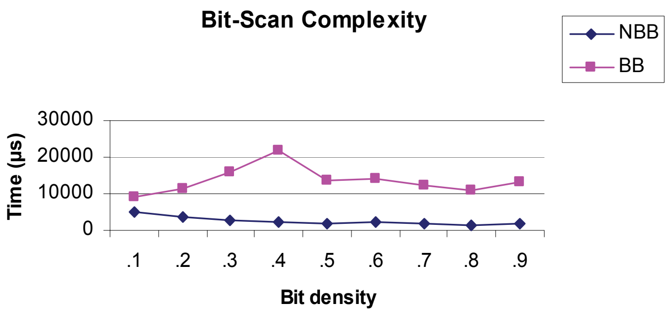


Figure 2. Different computing times for finding the first 100 elements in a randomly generated population of size 5000 after 1000 runs in a P-4 2.7GHz CPU. BB implements a compact bit array and NBB a list.

In the figure, BB stands for the compact bit array implementation, as opposed to a list or array made up of {0,1} integers. For the experiments the abovementioned 16 bit direct lookup table for LSB was employed. As expected, times for NBB remain reasonably linear with density whilst BB turns out to be more than 3 times slower in the general case. In (2), k=100, W_{size}=64, gives a 6 ratio difference in favour of NBB in the worst case, but average case for NBB is twice as fast since LSB will normally take two cycles and not four. As bit arrays become more and more sparse, average case for LSB decreases by another two factor since it takes one cycle to bit scan an empty block, so for d=0.1 the new ratio is $\frac{4 \cdot 100}{64 \cdot 4} \cong 1.5$.

Consider a bit vector space of states where a single bit vector BB_g guides some form of systematic search. This requires that every element of the set is expanded, so operator LSB_N must be called for every element of the set. Thus, the overall inherent complexity of the bit encoding is similar to finding the first k-bits in BB_g where k averages 1-bits for all states visited:

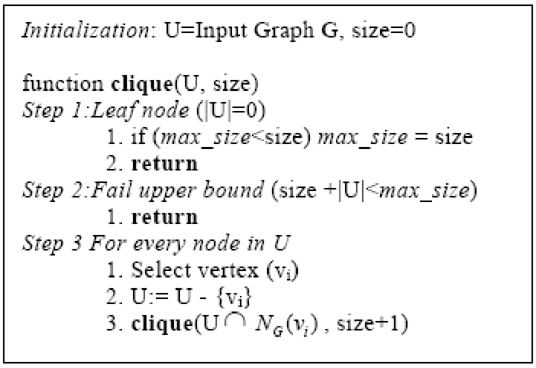
$$k = \frac{\sum PC(BB_g)}{Number\, of\;\; subproblems\;\, solved} \tag{3}$$

It is not clear that the benefits of computing transitions using bit-parallelism can outweigh this inherent bit scan complexity (e.g. in a brute force algorithm). In fact, the intuition is that additional bit encoded knowledge will be needed for efficient bit-parallel systematic search to improve a standard implementation. For some years now the authors have been interested in proving this statement for instances in the NP-complete class. Recent work in this line of research has led us to one such problem for the graph domain: the maximum clique problem. As a result, we have implemented a new complete bit-parallel general purpose algorithm which is one of the fastest general purpose algorithms at the moment (San Segundo P. et al, 2007). This result is important since it shows that bit-parallelism can be used as a tool to improve general purpose search algorithms for problems in NP. The following subsection focuses on this topic.

2.3 Bit encoded knowledge

The subsection is concerned with simple graphs. Simple graphs have a finite set of vertices V and a set E of pairs of vertices (x,y) called edges. Two vertices are said to be adjacent if they are connected by an edge. A subset of vertices such that every edge in W belongs to V is called a subgraph over G induced by W, and is written $G(E/W)$ or simply $G(W)$. A clique in G is an induced subgraph where every pair of vertices are bitwise adjacent. The *k-clique* problem consists in determining whether a clique of size *k* exists for a given graph and is well known to be NP-complete (Karp R.M. 1972). The corresponding optimization problem is the maximum clique problem (MCP), which looks for the largest possible clique in a given graph. MCP is NP-hard.

A typical efficient exact MCP algorithm uses a depth-first strategy to implement systematic search in a branch and bound scheme. Search takes place in a graph space starting with a small clique which gradually gets bigger and bigger as search advances. Recent examples of branch and bound algorithms for exact MCP are (Pardalos P.M. and Xue J. 1994) and more recent (Tomita E. and Kameda, T. 2006) amongst others. Figure 3 shows a primitive branch and bound MCP algorithm. It receives as input a simple graph *G* and returns the size of the largest possible clique in variable max_size. $N_G(v_i)$ is the neighbour set of vertex v_i in G and contains all vertices in G adjacent to v_i.



```
Initialization: U=Input Graph G, size=0

function clique(U, size)
Step 1:Leaf node (|U|=0)
        1. if (max_size<size) max_size = size
        2. return
Step 2:Fail upper bound (size +|U|<max_size)
        1. return
Step 3 For every node in U
        1. Select vertex ($v_i$)
        2. U:= U - {$v_i$}
        3. clique(U ∩ $N_G(v_i)$ , size+1)
```

Figure 3. A primitive exact branch and bound algorithm for MCP.

Simple graphs have {0,1} adjacency matrices where element A_{ij} is 1 if there is a corresponding edge between vertex *i* and vertex *j* in the graph and 0 otherwise. As a consequence, binary matrices map nicely to bit arrays of size the number of nodes of the graph (e.g. one bit array per row).
For a full encoding of the MCP search space, an additional bit array BB_{guide} is needed to guide the search, mapping the set of vertices of the graph at the current node. Initially BB_{guide} starts with all bits to 1 corresponding with the initial input graph. An empty BB_{guide} is a leaf node whilst vertices expanded in any path from root to leaf form a clique in *G* (see figure 4).

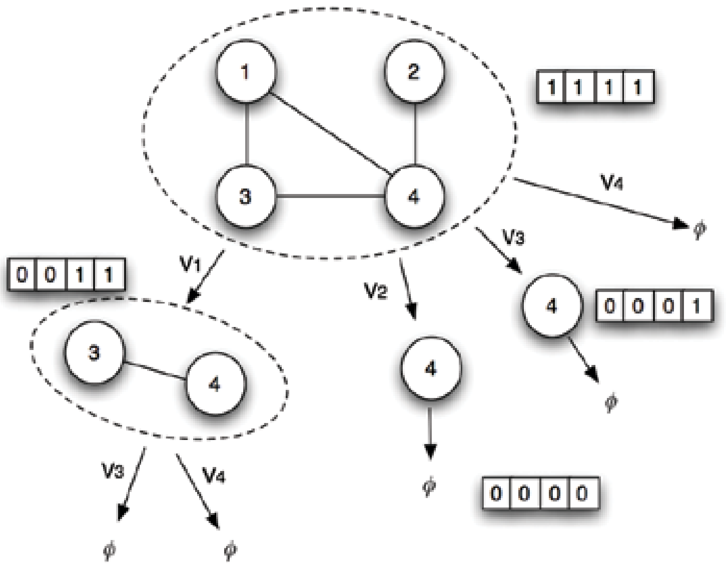


Figure 4. An example of MCP search in a bit encoded graph space. A single bit array guides the search. The bit encryption maps the i-th vertex of a graph to the i-th bit in the bit array. Every path from root to leaf node is a clique.

At every node bit scanning is needed during vertex selection for expansion, an overhead which has an important overall impact w.r.t. a non bit_parallel implementation. To validate this statement a number of tests have been carried out with a naïve brute force MCP algorithm denoted BBN-MCP (labelled BB in figure 5), and an equivalent non bit-parallel implementation N-MCP (labelled as *list* in figure 5). Both implementations use depth first systematic search to explore the full space without any pruning strategy and vertices are selected lexicographically.
Figure 5 shows time results for a number of randomly generated graph instances of different sizes and densities. In this systematic lexicographic brute force scenario, results indicate that the complexity of bit scanning at every node far outweighs the advantage of computing graph transitions efficiently using bitwise operations.
Things change when knowledge gathered during early exploration in depth-first search is bit encoded to prune the space later on. In MCP, strong efficient upper bounds on the size of the maximum clique for any graph can be computed through coloring of the graph vertices.

Classical vertex coloring of a graph $G=(V,E)$ is just a way to partition set V into disjoint subsets C_i of same color vertices. The restriction behind coloring is that only non adjacent vertices can be *painted* with the same color. Let C_i be the *i-th* color set of a possible *k-coloring* for G (see 4).

$$\bigcup_{i=1}^{k} C_i = V, \quad \bigcap_{i=1}^{k} C_i = \phi, \quad \omega(V) \le k \tag{4}$$

where $\omega(V)$ is common notation for the size of the largest possible clique in G. The best upper bound by vertex coloring for $\omega(V)$ is the graph chromatic number i.e., the minimum number of colors needed to paint the graph.

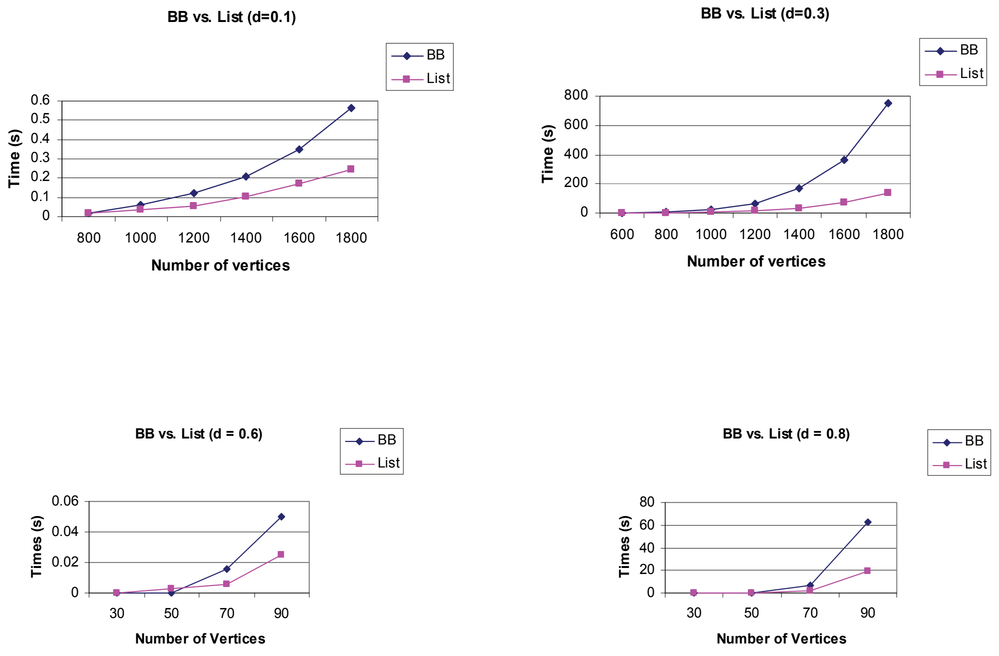


Figure 5. Time results for a bit-parallel (BB) and a classical (List) naïve brute force MCP algorithm.

Since optimal coloring is also in NP, efficient MCP complete algorithms use some form of greedy coloring strategy to prune the search space. There are many possible such strategies and an adequate survey is out of the scope of this article.

Of interest in this paper is the fact that previous naïve BBN-MCP implementation turns out clearly superior to N-MCP when a typical coloring scheme is added. The coloring implemented is a standard technique commonly used which is in $O(n^2)$, and runs w_{size} times faster in BBN-MCP than the non bit-parallel implementation. The impact of the pruning strategy for MCP is so big in the majority of cases that its computation becomes critical for overall efficiency. Figure 6 shows times for N-MCP and BBN-MCP when the coloring scheme is included. The situation is now reversed; bit scanning overhead is clearly surpassed by the benefits of bitwise coloring.

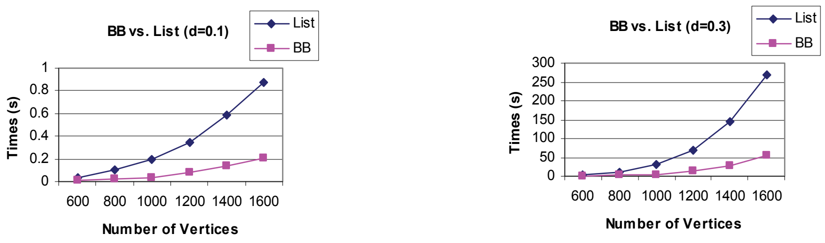


Figure 6. Time results for a bit-parallel (BB) and a classical (list) implementation of a naïve MCP algorithm with a classical vertex coloring strategy to establish bounds.

2.4 Graph models for bit-parallel search

The interest of this article is focused in efficient bit-parallel NP algorithms. In the authors' view, two very promising lines of research can be undertaken. In the first place, results presented in the previous subsection make MCP a promising tool for implementing bit-parallelism in other NP problems. A survey on our very efficient bit-parallel MCP algorithm can be found in our recent work (San Segundo P. et al. 2007). As has been said, *k-clique*, the corresponding non optimization version of MCP, is an NP-complete problem so it is certainly conceivable that problems with a reasonably benevolent reduction to *k-clique* can be efficiently solved using some form of bit-parallelism.

A second and more general line of research can be found in the intrinsic binary nature of simple graphs, which make them a very important tool by themselves to exploit bit-parallelism in search. The reason behind this is that the binary adjacency matrix of such graphs allows for a simple and clear mapping of relations to bits. Moreover it also facilitates the bit encoding of additional domain dependent knowledge, which can then be computed by efficient bitwise operations.

Following this second line of research, our attention has recently shifted to bit-parallelism in the Boolean satisfiability problem (commonly known as SAT). At the moment we have implemented a number of graph models to represent clause information with, as yet, modest but highly encouraging results. We note that today's fastest general purpose SAT algorithms do not employ reduction to a graph space; it is actually a more common practice to reduce other problems to SAT (e.g. (Kautz, H. and Selman, B. 1998) is a very efficient planner which solves a graph plan in a SAT space). Some NP-hard problems taken from board games have also an interesting reduction to simple graphs which might need reviewing from a bit-parallel perspective. One such example is N-Queens which aims to place N queens in an empty NxN square board such that they do not attack each other. More complicated scenarios include an initial non empty board (e.g. with a pawn on a particular square). A possible graph model for such scenarios maps vertices to squares in the board and places an edge between two squares if a queen placed on any one of them attacks the other.

Besides optimal search procedures we have also done some recent research on bit-parallelism in evolutionary algorithms. In this case the aforementioned intrinsic complexity of bit scanning is not necessarily a key issue because candidate solutions can be generated using other means (e.g. a permutation index). The issue of bit-parallelism in suboptimal search procedures is the focus of the next section.

3. Evolutionary algorithms

The term *Evolutionary Computation* denotes a class of population-based heuristic search techniques inspired by Darwin's principle of evolution in nature. Starting from a set of candidate solutions, called *population*, an evolutionary algorithm (EA) generates a new population of candidate solutions by means of operations called *selection*, *recombination*[4] and *mutation*, applied to the existing population. This step is called a *generation*. Generation after generation the population of candidate solutions *evolves* toward good solutions of the problems at hand. In analogy with natural environments, candidate solutions are also called *individuals*. Each individual is represented by a *chromosome*, which is an encoding of a candidate solution. Every individual has associated a *fitness* that is a measure of its quality, i.e., how good the individual is in solving the given problem. The term Evolutionary Computation denotes a whole family of techniques, which differ on some aspects from the evolutionary loop. Evolutionary Strategies, Genetic Algorithms, Genetic Programming, and many other evolutionary based search techniques apply the same basic concepts, but differ on how the selection, recombination, mutation, encoding of individuals and survivor selection operations are implemented. A detailed description of all the aspects of the evolutionary computation galaxy goes beyond the purpose of this work. For a good survey we refer the reader to (Eiben A.E. and Smith J.E., 2002).

Because of the natural representation of candidate solutions as string of bits, where each bit represents the truth value of the corresponding variable, SAT is the typical problem that can be approached using standard genetic algorithms (GAs), i.e. evolutionary algorithms based on bit-string representation of chromosomes. However, it was observed that since EAs do not use domain dependent knowledge, they may not outperform well tuned problem specific algorithms. This observation has been experimentally confirmed, and justifies the fact that all evolutionary algorithms for SAT proposed in the recent years have incorporated heuristic information. EAs for SAT can be roughly divided into three main classes depending on the way they use knowledge: EAs that encode knowledge into the fitness function, in the genetic operators and those that use the MAX-SAT fitness function (see below) and add local search to improve the quality of individuals. Usually EAs for SAT adopt the bit representation, since this is the most natural representation for this problem. However, EAs based on other representations have been used, like clausal representation, floating point, and path representation.

Several different evolutionary algorithms have been proposed for the SAT problem, varying in the representation and/or fitness function. For an exhaustive survey we refer to (Gottlieb et al., 2002). In the following, we will analyze the ASAP algorithm (*Adaptive evolutionary algorithm for the Satisfiability Problem*), which is one of the best evolutionary algorithms for the Sat problem, and proved to be competitive with the best non-evolutionary algorithms (Rossi, C. Et al., 2000).

The ASAP algorithm is a *(1+1)-evolutionary strategy* enhanced with a local search step and a memory of past states that is used to escape from local minima and to dynamically adapt the mutation parameter. In a *(μ+λ)-strategy* the population consists of μ individuals. At each generation, λ new individuals are generated, and the new population is formed by the best μ among the *(μ + λ)* individuals. In ASAP, since the population is formed by only one candidate solution, there is no recombination operator, and only mutation is used to

[4] Recombination is also known as *crossover*.

generate the offspring (see Fig. 7, left). Mutation consists in changing the value of a bit of the chromosomes chosen at random with a certain probability, called *mutation rate*.

3.1 Description of the ASAP algorithm

In ASAP, to each new individual a local search procedure called *Flip Heuristic* is applied. The technique of using local search operators in combination with evolutionary algorithms is called *Evolutionary Local Search* or *Memetic* search.

```
PROCEDURE ASAP
  randomly generate chromosome C
  apply Flip Heuristic to C
  WHILE (not termination condition) DO
    BEGIN
      C0=C      /* store parent C */
      apply adaptive mutation to C
      apply adaptive Flip Heuristic to C
      compute fitness of C
      ID(fitness C < fitness C0)
        C=C0   /* discard new C */
      ELSE
        UPDATE_TABLE(C)
    END
END PROCEDURE
```

```
PROCEDURE UPDATE_TABLE
  BEGIN
    IF (fitness C > fitness CO)
      BEGIN
        empty table T
        add C to table T
        unfreeze all genes
      END
      ELSE /* fitness CO=fitness C*/
        BEGIN
          add C to table T
          IF (table T full)
            BEGIN
              compute frozen genes
              adapt mutation rate
              empty table T
            END
        END
END PROCEDURE
```

Figure 7. ASAP pseudo-code

Roughly, it consists in the application of genetic operators to a population of local optima produced by a local search procedure. The Flip Heuristic consists in repeatedly flipping one bit in a randomly generated sequence, and keeping the change if this leads to an increment of the fitness function (i.e., more clauses becomes satisfied then becomes unsatisfied). When no increment has been obtained, the procedure terminates.
The choice of an appropriate fitness function is very important in the design of an evolutionary algorithm. ASAP adopts the most used fitness function for the Sat problem in EAs, called MAXSAT. The MAXSAT formulation assumes that the Sat problem is expressed in conjunctive normal form, i.e. it is a conjunction of *m clauses* c_i, $i=1..m$, each of which is a disjunction of *literals* (a variable or its negation).

$$f(x) = c_1(x) \wedge ... \wedge c_m(x), \quad c_i = (l_{i1} \vee ... \vee l_{ik})$$

where *x* is the array of the Boolean variables. In the MAXSAT formulation, the fitness value is equivalent to the number of satisfied clauses, i.e.,

$$f_{MAXSAT}(x) = val(c_1(x)) + . . . + val(c_m(x)),$$

where *val* $(c_i\ (x))$ maps the truth value of the i-th clause into an integer value 1 when the clause is true and 0 when it is false. In this way, the range of the function changes from {*true,false*} to {*0..m*}. Note that in this formulation the optimum value is known in advance, since the formula is satisfied when all its *m* clauses evaluates to 1.
ASAP is provided with a memory of past states. This is used to escape from local minima in a twofold way. Observe that at each generation the algorithm produces a local optimum.

Suppose the local search procedure directs the search towards similar (that is, having small Hamming distance) local optima having equal fitness function values. Then we can try to escape from these local optima by prohibiting the flipping of some genes and by adapting the probability of mutation of the genes that are allowed to be modified. To this aim, ASAP uses the following technique inspired on TABU search (see Fig. 7, right). At each step, a table T of size *k* is filled with chromosomes having best fitness. If the best fitness increases then the table is emptied. When the table is full, the chromosomes are compared gene-wise. Those genes that do not have the same value in all the chromosomes are labelled as *"frozen"*. The information contained in T is used for adapting the search strategy during the execution as follows. Each time T is full, the mutation rate is recomputed setting it to the value mutation_rate = ½ · (n^{o}. of frozen variables)/n (thus, 0 < mutation_rate < 0.5), and the flipping of frozen genes is prohibited. The rationale behind these two actions is the following. If table T becomes full it means that the search strategy has found for k times best chromosomes with equal fitness. A non-frozen gene has the same value in all these chromosomes. This indicates that the search directs often to local optima containing the same values of such genes. Therefore in the next iteration we allow to flip only not frozen genes in order to reach search points far enough from the attraction basin of those local optima. The mutation rate is chosen in such a way that the lower the number of not frozen genes is, the higher the probability will be to flip them, since a strong basin of attraction, requires a higher probability of generating individuals that are very different ("far") from its parent. The term 1/2 is used to keep the mutation rate smaller than or equal to 0.5.
Although the most obvious way to represent a solution candidate for SAT is a bit string of length n, where every variable is associated to one bit, in the original implementations of ASAP this was for simplicity encoded as an array of integer values, taking the value 0 or 1. A new version of ASAP has been implemented adopting the bit board representation, in order to exploit the benefits of bit-parallelism. We will refer to the new version as ASAP-BB. In order to analyze the benefits of adopting the new representation let us analyze in detail the computational cost of producing a new solution ASAP.

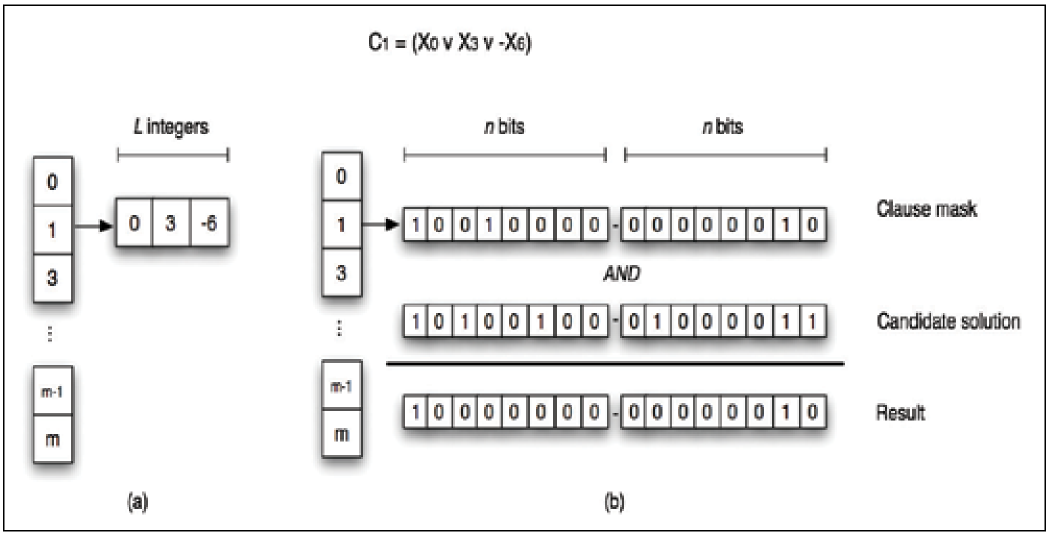


Figure 8. Clause representation: (a) array of indexes; (b) bit arrays.

Let m be the number of clauses, n the number of variables and L the average clause length (number of literals) of a given SAT instance. As mentioned before, each time an offspring candidate solution is generated, it is repeatedly improved by a series of bit flips. Each time a bit is flipped the fitness function must be recomputed in order to check whether the change leads to an improvement. This is the most expensive operation and is repeated several times. Computing the fitness function implies looping through the clauses and re-computing them by assigning to its literals the value of the corresponding variable of the solution. In ASAP, a solution is represented as an array of n integers, and a clause is represented as an array of integer values, containing the indexes of the variables contained in the clause (see Fig. 8 (a)). Thus, on average, each clause computation involves L integer operations. The cost of a fitness function evaluation is $m \cdot L$.

In ASAP-BB, a solution is encoded as a bit vector of length $2n$, containing the values of the variables in the first half, and their negation in the second half. A clause is encoded as a bit vector of $2n$ bits, called clause mask. The first n bits have their value set to '1' if its position corresponds to the index of a non-negated variable of the clause, and '0' otherwise. The second n bits are set in the opposite way: bits are set to '1' in correspondence of negated variables (see Fig. 8 (b)). Thus, the evaluation of a clause is performed as a bit-wise *AND* operation between a solution and the clause mask which allows exploiting bit parallelism. The cost W of such operation depends on the word length and the number of variables of the problem at hand:

$$W = N\_WORDS = \left\lceil \frac{2n}{WORD\_LENGTH} \right\rceil \tag{5}$$

and the total cost of evaluating the fitness function will be $m \cdot W$.

Thus, considering that the fitness calculations are the core of the algorithm, and everything else is kept unchanged, the expected speedup of ASAP-BB w.r.t ASAP is

$$Speedup = \frac{mLF}{mWF} = L / W \tag{6}$$

where F is the total number of fitness evaluations. The expected speedup depends on the average clause length and on the number of variables, the latter determining the size of the bit array.

As far as space is concerned, a similar analysis can be performed. The total storage space for a clause in ASAP is $L \cdot m$ integers while in ASAP-BB it is $2 \cdot n \cdot m$ bits. Assuming an integer has a size of four bytes, the space occupation ratio is

$$Space = \frac{2mn/8}{4mL} = \frac{n}{16L} \tag{7}$$

3.2 Experiments

In order to validate the previous analysis of time and space complexity, we have performed a series of tests on a set of standard benchmark instances, all satisfiable. Instance family

3SAT was the first used to test different EA-based algorithms for SAT (cf. Bäck T. et al., 1998). These instances are random 3-SAT benchmark instances with *m/n = 4.3*[5] generated using the *mkcnf*[6] generator using the *forced* option to ensure that they are satisfiable. Instance families *II, Aim, Jnh, Par* are taken from the 2nd DIMACS challenge on cliques, coloring and satisfiability (Johnson D. and M. Trick, 1996). The *Aim* family contains artificially generated 3-SAT instances and are constructed to have exactly one solution. Family *Par* instances arise from a problem in learning the parity function. The *Jnh* instances are randomly generated and have a varying clause length. Instances *II* arise from the "Boolean function synthesis" problem and are used in inductive inference.

Table 2 reports the results of the tests performed. In order to compute the real speedup, times for ASAP and ASP-BB are averaged after 10 runs on every instance[7]. The speedup values have been computed averaging all the results of instances with similar properties (i.e. *m* and *L* values).

The table shows that the measured speedup is in accordance with the analysis performed, with small differences that are, in general, smaller than the standard deviation σ, and thus are not statistically significant. Note that the Par and II instances have a clauses/variables ratio that is disadvantageous for the bit array representation.

As far as the space ratio is concerned, the bit vector representation saves space w.r.t. the plain integer array representation only when the number of literals remains low. Worst case space ratio occurs for II instances, with a 70% increment approx.

4.The geometric correspondence problem

In this final section we present a survey on recent work done by the authors where bit-parallelism has been applied to a real life problem with success. More specifically, an exact bit parallel algorithm for the maximum clique problem has been conveniently applied to solve the correspondence problem between two sets of geometric entities, also known as relational structure search (Bomze et al., 1999) in the vision domain or the data association problem (Siegwart & Nourkbash, 2004) in mobile robotics. The section starts with a description of an adequate representation of the problem for reduction to MCP and ends with some experiments with real data.

4.1 Description

Given two sets of geometric features (i.e. points, segments etc.) the aim is to find the best correspondence between both sets. If a weighted graph of geometric relationships is built in each set, with a relationship (e.g. a metric) established between every two features, the problem becomes that of finding the Maximum Common Subgraph (MCS) between them. MCS is known to be NP-hard, and its solution becomes even more difficult in the case of noisy scenarios, when simplifying hypothesis or approximations cannot be applied. This

[5] The 4.3 clause/literal ratio is such that instances generated with lower ratio (the *underconstrained* region) almost always have solutions. Those generated with higher ratio (the *overconstrained* region), almost always have no solutions. Recent works have identifed that hard random k-SAT instances lie in such backbone, also know as *phase transition* region.

[6] See ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/contributed/UCSC/instances

[7] ASAP is non-determinist, hence multiple runs must be performed in order to obtain an average behaviour.

setting occurs very often in many applications, as comparing fingerprints, mobile robot global localization, computer vision, pattern matching, etc.

Family	*Instance*	*No. of variables (n)*	*No. of clauses (m)*	*Average clause length (L)*	*BB length (W)*	*Expected speedup*	*Average speedup*	*σ*	*Space ratio*
3SAT	1	30	129	3.00	1	3.00	2.959	0.092	0.63
	2	30	129	3.00	1	3.00			
	3	30	129	3.00	1	3.00			
	4	40	172	3.00	2	1.50	1.551	0.178	0.83
	5	40	172	3.00	2	1.50			
	6	40	172	3.00	2	1.50			
	7	50	215	3.00	2	1.50	1.446	0.145	1.04
	8	50	215	3.00	2	1.50			
	9	50	215	3.00	2	1.50			
Aim	50-3_4-1	50	170	3.00	2	1.50	1.497	0.022	1.04
	50-3_4-2	50	170	3.00	2	1.50			
	50-3_4-3	50	170	3.00	2	1.50			
	50-3_4-4	50	170	2.99	2	1.49			
	50-6_0-1	50	300	3.00	2	1.50			
	50-6_0-2	50	300	2.99	2	1.50			
	50-6_0-3	50	300	2.99	2	1.50			
	50-6_0-4	50	300	3.00	2	1.50			
II	8a1	66	186	2.42	3	0.81	0.870	-	1.70
Jnh	1	100	850	5.17	4	1.29	1.311	0.014	1.21
	201	100	800	5.19	4	1.30			
	12	100	850	4.91	4	1.23	1.438	0.015	1.27
	204	100	800	4.89	4	1.22			
	205	100	800	4.89	4	1.22			
	210	100	800	4.89	4	1.22			
	213	100	800	4.88	4	1.22			
	218	100	800	4.88	4	1.22			
	7	100	850	4.89	4	1.22			
Par	8-1-c	64	254	2.88	2	1.44	1.410	-	1.39
	8-2-c	68	270	2.89	3	0.96	0.930	0.033	1.47
	8-3-c	75	298	2.90	3	0.97			1.62
	8-4-c	67	266	2.89	3	0.96			1.45
	8-5-c	75	298	2.90	3	0.97			1.62

Table 2. Results for SAT tests using ASAP and ASAP-BB. Times are averaged after 10 runs.

The relationships between geometric features, also called constraints, are pose invariant relationships that relate both features. If the sets are composed by 2D features, the constraints could be:

- **Distance**: Euclidean distance between two points.
- **Angle**: Angle between two non parallel segments.
- **Distance**: Shortest (perpendicular) distance from point to segment.
- **Distance**: (Shortest) distance between two parallel segments.

A weighted graph reduction to MC capturing such relationships in each set would map vertices to the elements of the set and weighted edges to the constraints.
This geometric correspondence problem allows a more convenient formulation (Bailey 2002) under the MCP paradigm. Figure X illustrates this idea employing two sets L and O. The former contains N features called landmarks (L_1,…, L_n), and the latter is the observation set containing M features called observations (O_1,…, O_m). Instead of searching the MCS between those two sets, a new graph called the association graph is defined, in which the nodes represent each possible landmark-observation pairing. Thus, the number of vertices of the correspondence graph is a priori NxM.

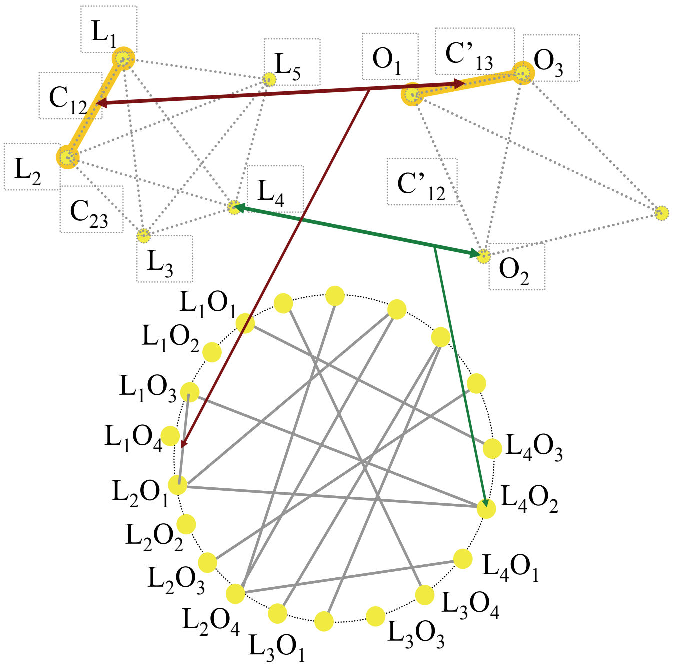


Figure 9. The geometric correspondence problem as an MCP search in an associations graph

The edges of the association graph are defined by checking pairs of constraints between the landmark and the observation initial graphs. If a constraint that relates two landmarks in the landmark set (e.g. constraint C_{12} relates landmarks L_1 and L_2) is compatible with a constraint that connects two observations in the observation set (in the example C_{13}' that connects O_1 and O_3), then L_1O_3 and L_2O_1 vertices in the association graph are connected. In this new association graph, the correspondence problem is reduced to finding the maximum clique, equivalent to maximizing joint compatibility. Once the problem has been reduced to MCP we have applied bit-parallelism in a similar way as described in section 2.

4.2 Experiments

Two different sets of experiments with the proposed solution to the geometric correspondence problem have been carried out: image matching and mobile robot global localization. In both scenarios, the required processing time has been the main output. Solutions obtained in all cases are optimal. The results have been compared with a finely tuned version of the MCS algorithm, running in the same computer.

4.2.1 Image matching

In this experiment, a large aerial image of our city, Madrid has been selected because its repetitive structure. The total image size is 1806x1323 pixels, and each pixel represents approximately 0,4m, so the area covered is about 792x580m. Figure X shows one ninth of such image and it can clearly be observed that its "texture" is quite repetitive making the recognition of a partial image hard for the human eye. Given a partial subimage with unknown position and orientation, the problem studied is to find the correspondence in the full image. Pre-processing includes corner detection as in (Rosten & Drummond, 2005, Rosten & Drummond, 2006) applied to both images to extract relevant points that can be used as features. It has to be noted that due to different lighting conditions, noise, dynamic objects, not necessarily the same corners are detected in both images (see figure X). In such a noisy scenario, the exact solution could be the only way to guarantee robustness.

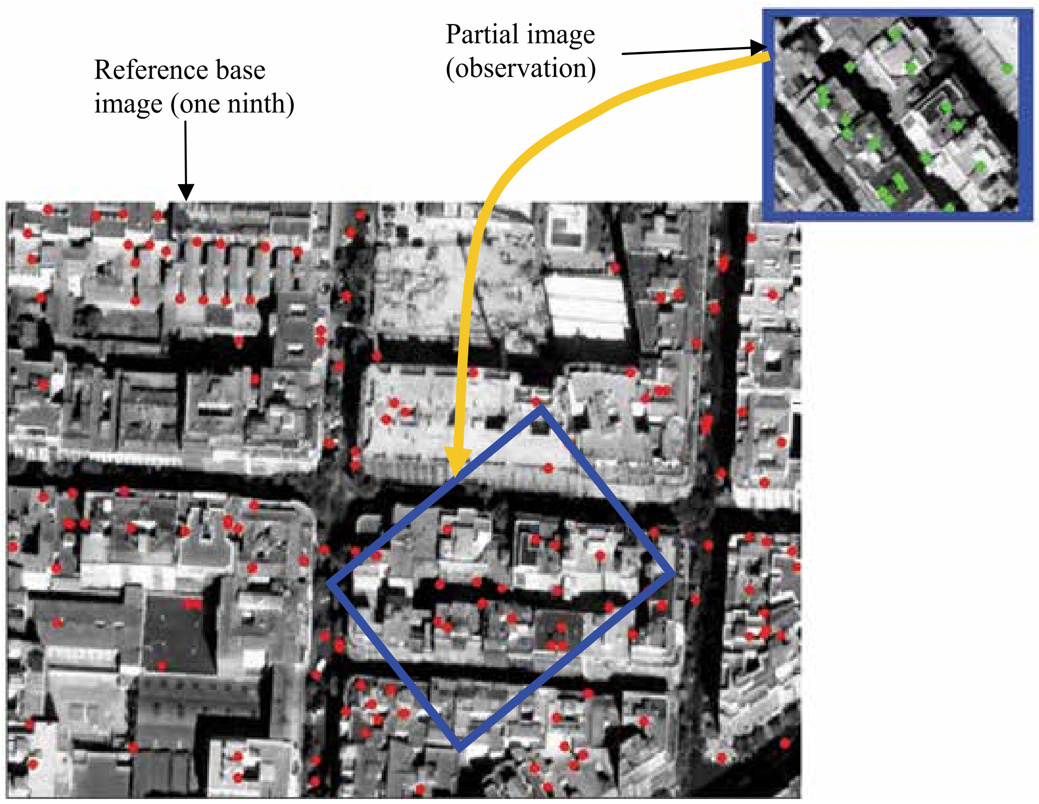


Figure 10. Computer vision pattern matching

The table shows that processing time quickly increases with the number of landmarks and observations for the MCS algorithm, but BE-MCP remains reasonably low. Furthermore, the variance of BE-MCP is quite small w.r.t. MCS, which increases rapidly with the size of the problem.
Table 3 shows processing times for both algorithms in different settings of landmark-observation pairs, depending on the corner detection threshold chosen. Each setting is repeated 10 times with different random observation subimages, and the average, best and worse times are shown in the table. The algorithms were implemented in C++ and tests were run on a P4 2,6GHz laptop.

Time (s)	Algorithm	MCS				BE-MCP			
	Number of landmarks	1023		1464		1023		1464	
Number of observations	7	3,22	8,83	16,24	31,75	0,09	0,09	0,18	0,20
			0,55		4,67		0,06		0,17
	10	10,89	23,28	41,44	87,45	0,16	0,17	0,35	0,36
			4,17		14,66		0,14		0,33
	15	32,83	59,70	86,04	207,25	0,37	0,39	0,79	0,95
			6,81		28,64		0,36		0,74

Table 3. Comparison of processing times for different settings between a bit-parallel MCP search algorithm and an MCS solver

4.2.2 Mobile robot global localization

Finding a robot position in a given map with only observations on local features is called mobile robot global localization. If the map contains a set of geometric entities such as segments (e.g. the environment walls), the observations of the robot will also be modelled as such, but due to noise, dynamic objects (ie.g. people) and sensor limitations, these observations can be also noisy and incomplete.
Using the MCP approach we have solved the global localization problem using real data from our interactive tour-guide robot called Urbano (see figure 11). The reference maps were built in real time with an EKF based SLAM algorithm (Rodriguez-Losada et al. 2006). This time comparisons between BE-MCP and MCS were carried out under different levels of noise, in a map composed by approximately 350 features, with observation sets made up of between 25 and 30 observations. In the observation sets, a variable number of spurious observations were allowed ranging from 5 to 14 (the latter being almost 50% of the total).
In this case the average time required for MCS increased with the number of noisy observations, but the time required for BE-MCP remained constant. Furthermore, the variance in the BE-MCP also remained constant, while the variance in MCS increased degrading the worst case performance.

5. Conclusions and future work

Using bit-parallelism to implement efficient search algorithms raises a number of fundamental questions which the authors have tried to cover to some extent in this chapter. In the first place it has been shown that scanning for 1-bits in a compact bit arrays is an intrinsic overhead which must be taken seriously into account, especially in systematic search procedures. In the second place emphasis has been laid on the importance of simple graph models because of their inherent binary adjacency matrix-bit array mapping. This fact ensures a natural bit encoding of frame knowledge as well as facilitates bit encoding of additional domain dependent knowledge. Following this line of research, recent work done by the authors on the maximum clique problem has revealed that its particular nature makes it a very good tool to implement efficient bit parallel algorithms for problems in NP.

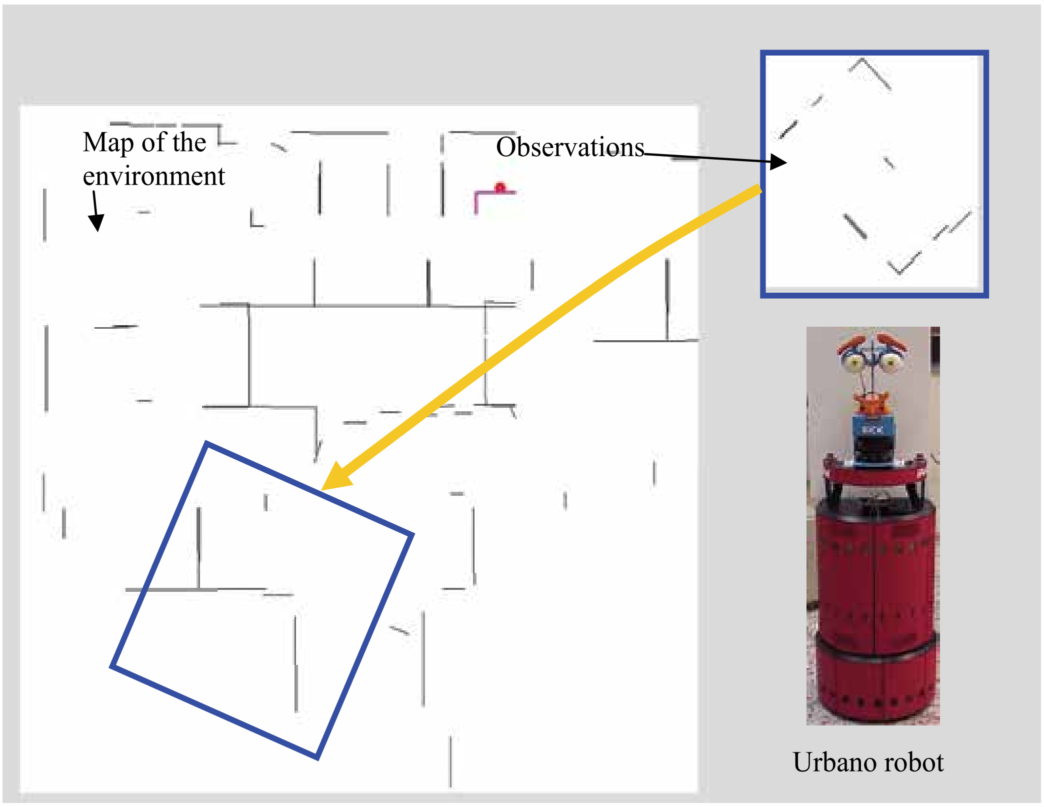


Figure 11. Mobile robot global localization

Attention has also been paid to bit-parallelism in suboptimal search. The analysis and experiments with the ASAP genetic algorithm have shown that bit parallelism can be beneficial for the SAT problem depending on the problem instance and of the specific data structures used to manage the bits. An optimized bit array structure would allow achieving even better performances than the ones obtained in the experiments performed here.
Finally the authors present a brief survey on two real life applications where bit-parallelism has proved successful.

6. Acknowledgements

This work is funded by the Spanish Ministry of Science and Technology (Robonauta: DPI2007-66846-C02-01) and supervised by CACSA whose kindness we gratefully acknowledge.

7. References

Rosten E. and Drummond T., 2005. Fusing points and lines for high performance tracking. IEEE International Conference on Computer Vision. Oct 2005. Vol 2. pp 1508-1511.

Rosten E. and Drummond T., 2006. Machine learning for high-speed corner detection. European Conference on Computer Vision.

Rodriguez-Losada D., Matia F., Galan R. Building geometric feature based maps for indoor service robots. Elsevier: Robotics and Autonomous Systems. Volume 54, Issue 7 , 31 July 2006, Pages 546-558

Bomze I.M., Budinich M., Pardalos P.M., Pelillo M., 1999. HandBook of Combinatorial Optimization, Supplement Vol A. Kluwer Academic Publishers, Dordrecht, 1999 pp.1-74.

Bailey T. 2002, Mobile Robot Localisation and Mapping in Extensive Outdoor Environments. PhD thesis. Australian Centre for Field Robotics, University of Sydney.

Siegwart R., Nourkbash. I. An Introduction to Autonomous Mobile Robots, MIT press, 2004.

Eiben A.E. and Smith J.E., 2002. Introduction to Evolutionary Computing, Springer, 2002.

Gottlieb J., Marchiori E. and C. Rossi 2002, Evolutionary algorithms for the satisfiability problem. Evolutionary Computation Vol. 10, Nr. 1, pp. 35-50, 2002.

Rossi C., Marchiori E. and Kok J.N. 2000, An adaptive evolutionary algorithm for the satisfiability problem. In Proceedings of ACM Symposiumn Applied Computing, pages 463–469, 2000.

Bäck T., Eiben A.E. and M. Vink 1998 A superior evolutionary algorithm for 3-SAT. In Saravanan, N., Waagen, D., and Eiben, A., editors, Proceedings of the Seventh Annual Conference on Evolutionary Programming. Lecture Notes in Computer Science, Volume1477, pages125–136, Springer, Berlin, Germany, 1998.

Johnson D. and M. Trick editors 1996, Cliques, Coloring and Satisfiability. AMS, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 26, 1996.

Baeza-Yates R. and Gonnet G. H 1992, A new approach to text searching. Commun. ACM, 35(10), pp:74–82, (1992).

Heinz E.A. 1997, How DarkThought plays chess, ICCA Journal, 20(3): pages 166-176, 1997.

San Segundo P., Rodriguez-Losada D., Galán R., Matía F. and Jiménez A. 2007, Exploiting CPU bit parallel operations to improve efficiency in search. International Conference on Tools for Atificial Intelligence (ICTAI 07). Patrás, Grecia, Octubre 29-31, 2007.

Leiserson, C., Prokop, H., and Randall, K. (1998). Using de Bruijn sequences to index a 1 in a computer word. See: http://supertech.csail.mit.edu/papers/debruijn.pdf.

Karp R.M. 1972. Reducibility among Combinatorial Problems. Editors: R.E. Miller, J. W. Thatcher, New York, Plenum, pp: 85-103 (1972).

Pardalos P.M. and Xue J. 1994, The maximum clique problem. Global Optimization. 4: pp. 301-328, (1994).

Tomita E. and Kameda, T. 2006. An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. Journal of Global Optimization (37), Springer, pp: 37:95-111 (2006)

Kautz, H. and Selman, B. 1998. BlackBox: A new approach to the application of theorem proving to problem solving. En AIPS98 Workshop on Planning as Combinatorial Search, pag. 58-60, Junio 1998.

Warren H.S. Jr 2002, Hacker´s Delight. Addison-Welsey 2002.

Multi-Sensor Fusion for Mono and Multi-Vehicle Localization using Bayesian Network

C. Smaili[1], M. E. El Najjar[2], F. Charpillet[1] and C. Rose[1]
[1]*LORIA-INRIA Lorraine- MAIA Team Campus Scientifique,*
[2]*LAGIS-CNRS UMR 8146 Polytech'Lille,*
France

1. Introduction

Outdoor mobile robotised vehicles currently hold the attention of many researchers because they can bring solutions to many applications related to transport of passengers in urban environments. An example of robotised vehicle is the CyCab (Bom et al., 2005). Transport applications which can combine mono or multi vehicle operating mode are most efficient. For autonomous navigation application, the vehicle needs to know its position accurately (Dissanayake et al. 2001); (Thrun et al. 2001) and if possible on the road network. In this work, we propose to use the digital road map database for the geo-localization of the vehicle. The localization of a vehicle using or in respect to or on a road map is treated by several ways in the last ten years. This relatively recent research theme is known also as the map-matching or road-matching problem. It can be interesting to localize a vehicle using a road map because it can useful to recover the attributes associated with these data bases. Examples of attributes are the width of the road, the presence of landmarks for accurate localization, authorized maximum speed for advanced driver assistance system application etc. Unfortunately, the use of the road map to improve the localization is not a simple task. There are always errors on the estimate of the position and because the map can represents a deformed sight of the world.
Outdoor positioning systems often rely on GPS, because of its affordability and convenience. However, GPS suffers from satellite masks occurring in urban environments, under bridges, tunnels or in forests. GPS appears then as an intermittently-available positioning system that needs to be backed up by a dead-reckoning system (Zhao, 1997); (Abbott & Powell, 1999); (EL Najjar & Bonnifait, 2003). In this work, the proposed method of multi-sensors fusion for mono-vehicle localization is based on the use of encoders positioned at the rear wheel of the vehicle. We use these sensors to measure elementary rotations of the wheels and to estimate the displacement of the vehicle. Thus, a dead-reckoned estimated pose is obtained by integrating the elementary rotations of the wheels using a differential odometric model. The multisensor fusion of GPS and odometry is performed by a Bayesian Network (BN).
Afterwards, we extend the multi-sensor fusion method proposed in this work for mono-vehicle localization to be used for the localization of several vehicles moving in the same environment. We suppose in this extension that the vehicles evolve in a train configuration. In the literature, we found two ways to make moving a train of vehicles. The first one

proposes to use an inter-vehicles communication (Bom et al., 2005). The second approach proposes to not use any communication support and to make moving vehicles near-by-near (Daviet & Parent). In our approach, we assume that the leader vehicle is equipped by accurate positioning sensors (GPS LRK, gyroscope LASER and road map database) and multi-sensors fusion approach in order to have continuous and accurate geo-position information. Followers' vehicles are equipped by relatively low cost proprioceptifs and exterioceptifs sensors (odometer, Lidar[1], DGPS) for localization task. The path of the leader vehicle is assumed to be propagated to followers using an inter-vehicle communication device.
The paper is organized as follows: section 2 describes the architecture of vehicle localization. In Section 3, we propose an overview of Bayesian networks formalism. Then, we describe a BN model for the localization of mono-vehicle and the extension of the method in same formalism for the localization for multi-vehicle in section 4. Finally, real data results and simulation are presented and analyzed.

2. Architecture of vehicle localization method

The vehicle localization method described in this section relies on Bayesian networks. The proposed approach can be described by Fig.1. Firstly, the algorithm combines the Anti-lock Braking System (ABS) measurements with a GPS position, if it is available. Then, using this estimate, segments around the estimation are selected in a radius of 30 meters by using a Geographical Information System (GIS-2D). Using these segments, map observations are built and merged with other data sensors using a method based on Bayesian network.

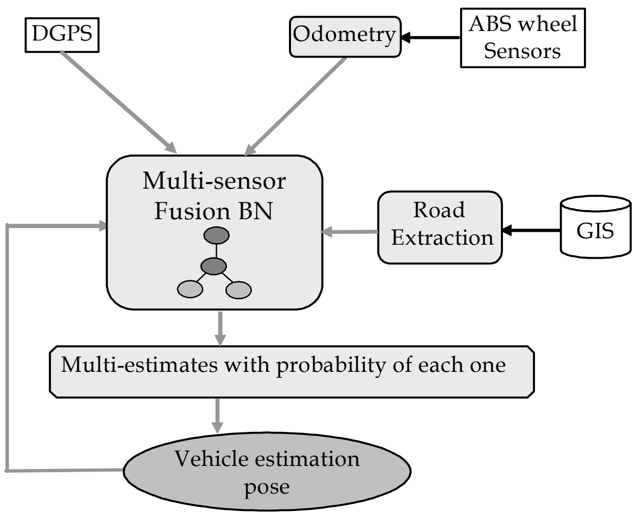


Fig.1. Synoptic of the Mono-vehicle localization method

2.1 Localization and heading estimation by combining odometry and GPS

Let us consider a car-like vehicle with front-wheel drive. The mobile frame is chosen with its origin M attached to the centre of the rear axle. The x-axis is aligned with the longitudinal

[1] Lidar is a SICK LASER telemeter.

axis of the car (see Fig.2). The vehicle's position is represented by the (x_k,y_k) Cartesian coordinates of M in a world frame. The heading angle is denoted θ_k. If the road is perfectly planar and horizontal, and if the motion is locally circular, the motion model can be expressed as (EL Najjar & Bonnifait, 2005):

$$X_{k+1} = \begin{cases} x_{k+1} = x_k + d_s.\cos(\theta_k + \frac{\omega_\theta}{2}) \\ y_{k+1} = y_k + d_s.\sin(\theta_k + \frac{\omega_\theta}{2}) \\ \theta_{k+1} = \theta_k + \omega_\theta \end{cases} \tag{1}$$

Where d_s is the length of the circular arc followed by M and ω_θ is the elementary rotation of the mobile frame. These values are computed using the (ABS) measurements of the rear wheels. Let denote X_{k+1} the state vector containing the pose.

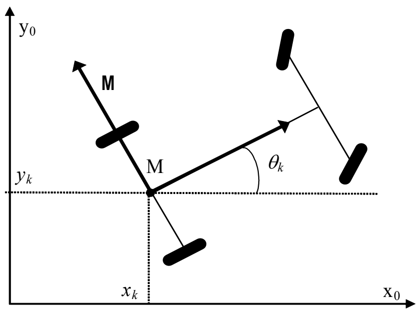


Fig.2. The mobile frame attached to the car

2.2 Cartographical and GPS observation equation

The selection of candidate roads is the first stage of the localization on a road map problem. Generally, this involves applying a first filter which selects all the segments close to the estimated position of the vehicle. The goal is then to select the most likely segment(s) from this subset. Nowadays, since the geometry of roadmaps is more and more detailed, the number of segments representing roads is increasing. The robustness and complexity of the localization depends mainly on the road selection module. In order to be focused on this point, an accurate map Géoroute V2 provided by the French National Institute of Geography (IGN) was used in this work. Our strategy is based on the fusion of several criteria using distance direction measurements within the framework of Bayesian Network. The pose obtained by GPS and odometry can be more accurately estimated by fusing the selected segment. The key idea is to model the fact that the true position of the vehicle is located around the centreline of the most likely road. This region depends mainly on the width of the road, which is an attribute also stored in the database. We suggest using the most likely road in order to build a new observation with its estimated associated error.
In practice, when the GPS satellites signal is blocked by buildings or tunnels, the odometric estimation is used to select the segments all around the estimation from the cartographical database. The cartographical observations can be obtained by projections onto the segments. If the orthogonal projection onto line does not make part of the segment, the closer

extremity is used. When several segments are candidates, the cartographical observation function is a non-linear multi-modal observation. Considering a Gaussian distribution of noise to represent the uncertainty zone all around a segment, so the multi-modal observation is a multi-Gaussian observation one. The observation equation of the segment seg$_i$ can be written:

$$Y_{carto}^{segi} = \begin{bmatrix} x_{carto} \\ y_{carto} \\ cap_{carto} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \beta_{carto} \quad (2)$$

Where (x_{carto}, y_{carto}) is the projection onto each segments and cap$_{carto}$ is the segment heading. To represent the error of the cartographical observation, we choose a Gaussian distribution of the uncertainty zone all around the segment. So this error can be represented with an ellipsoid which encloses the road. This ellipsoid has its semi-major axis in the length of the segment and its semi-minor axis equals to the width of the road (EL Najjar & Bonnifait, 2005) (see Fig.3).

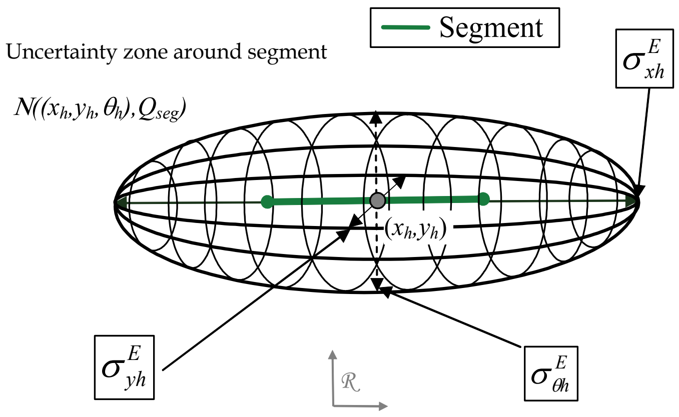


Fig.3. Ellipsoidal of probability construction representing zone around a segment for horizontal segment i.e. parallel with the east axes

The third axis of the ellipsoid represents the uncertainty of the estimation of the segment. This uncertainty is related to the relative error of the cartographical database. The covariance matrix of the cartographical observation error can be written:

$$Q_k^{carto} = \begin{pmatrix} \sigma_{x,h}^2 & Q_{xy,h} & 0 \\ Q_{xy,h} & \sigma_{y,h}^2 & 0 \\ 0 & 0 & \sigma_{\theta,h}^2 \end{pmatrix} \quad (3)$$

The GPS position measurement provides the GPS observation (x_{gps}, y_{gps}). The GPS measurement error can be provided also and in real time using the Standard NMEA sentence "GPGST" given by the Trimble AgGPS132 receiver which has been used in the experiments. The covariance matrix of the GPS error can be expressed as:

$$Q_k^{gps} = \begin{pmatrix} \sigma^2_{x,gps} & Q_{xy,gps} \\ Q_{xy,gps} & \sigma^2_{y,gps} \end{pmatrix}_k \tag{4}$$

The observation equation can be written:

$$Y_{GPS} = \begin{bmatrix} x_{gps} \\ y_{gps} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \beta_{gps} \tag{5}$$

3. Bayesian networks

A Bayesian network can be defined as a pair G= (S, Φ), where S is a directed acyclic graph and Φ is a parameterization of a set $\{P(X_1|\Pi_1),\ldots, P(X_n|\Pi_n)\}$ of conditional probability distributions, one for each variable, and Π_i is the set of parents of node X_i in S. Being directional and acyclic the structure of a Bayesian network provides a direct factorization of the joint probability distribution as (Castillo et al., 1997):

$$P(X_1,\ldots, X_n) = \prod_{i=1}^{n} p(X_i \mid \Pi_i)) \tag{6}$$

The notation used in this work is adopted from (Cowell et al., 1999), (Murphy 2002) and (Murat 2001) where round nodes were used to denote continuous random variables and square nodes denote discrete random variables. Throughout this paper, X_i denotes a continuous or discrete random variable. Values of the random variable will be indicated by lower case letters as in x_i. For a discrete variable that take r values, x_i^k denote a specific assignment for 1≤ k ≤ r. A set of variables is denoted in boldface letters **X**=$\{X_1,\ldots,X_n\}$.

3.1 Inference engine

An important issue in Bayesian networks is computation of posterior probabilities of some variables given observation. The inference problem in general Bayesian networks is still a hot research topic (Heckerman 1995). Several researchers have developed exact and approximate inference algorithms for different distributions (Murphy 2002). The most commonly used exact inference algorithm for discrete Bayesian networks is known as the JLO algorithm (Jensen et al., 1990). The JLO algorithm is a recursive message passing algorithm that works on the junction tree of the Bayesian network. The junction tree is constructed from the directed acyclic graph using some graph-theoretic tools. In the following we propose to detail an example of a discrete Bayesian network. A complete definition using continues and hybrid Bayesian networks are given in (Cowell et al., 1999); (Jensen, 2001) and (Murphy, 2002).

3.2 Constructing the junction tree

In order to define the junction tree we need to define the term clique. A clique is a complete set of nodes which is not a proper subset of another complete set (Castillo et al., 1997). A set of nodes is said to be complete if every pair of nodes in the set is linked. Fig.4 (a) contains the following cliques: C_1={A,B}, C_2={B,C}, C_3={C,D}, C_4={D,H}, C_5={D,E,G}, C_6={E,F,G} and

C_7={A,E}. However, if we add some extra links to the graph, some of the previous maximal complete sets are no longer maximal, and the graph contains different cliques. For example, the graph in Fig.4 (b) is obtained by adding three links to the graph in Fig.4 (a). The sets C_1, C_2, C_3, and C_7 are no longer complete. Thus, the graph in Fig.4 (b) contains five cliques: C_1={A,B,D,E}, C_2={B,C,D}, C_3={D,H}, C_4={D,E,G}, and C_5={E,F,G}.

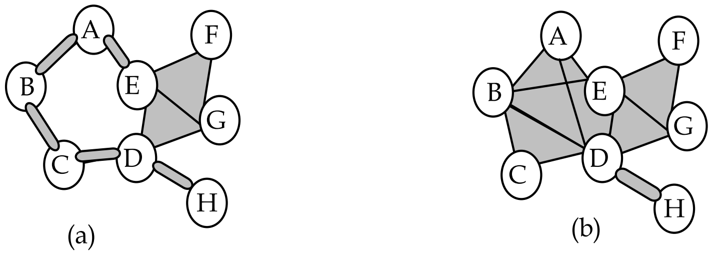


Fig.4. Examples of cliques associated with two different graphs

A junction tree is a tree of cliques that satisfies the running intersection property (RIP). RIP implies that if a node is contained in any two cliques, then it is contained in all the cliques in the unique path between them (see example in 3.2.3).

3.2.1 Moralization

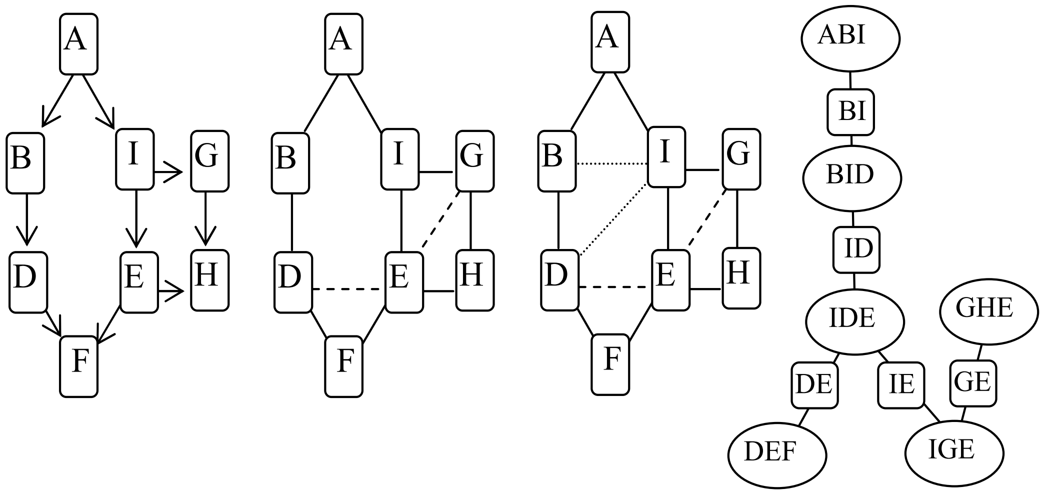
The moral graph of a directed acyclic graph is obtained by introducing additional undirected edges between any two nodes with a common child and subsequently replacing all directed edges with undirected ones (see Fig. 5 (b)). The moralization process guarantees that a family of nodes (a node with all of its parents) will occur together in a clique (Castillo et al., 1997).

3.2.2 Triangulation

The second step to construct a junction tree is to add sufficient edges to the moral graph to obtain a triangulated (chordal) graph. The aim of triangulation is to obtain a decomposable model such that the joint probability distribution can be factorized over the clique potential function (Cowell et al., 1999). An undirected graph is triangulated if every loop of length four or more $X_1 - X_2 - \ldots - X_{n-1} - X_1$ has at least one chord i.e. a link $X_i - X_j$ between two non-consecutive nodes X_i and X_j. If the required chords are not already in the set of edges, they are added, in order to get a triangulated graph. The triangulation process is not unique. In general it is desired to obtain a triangulation with a minimum number of additional edges. The additional edges have a major influence on the time complexity of the inference process (Yannakakis, 1981); (Kjaerulff, 1990). Fig.5 (c) is a trivial example for a triangulated graph. The moral graph (Fig.5 (b)) has a loop of length six: A—B—D— E— I—A. To get a triangulated graph, the links B—I and I—D can be added. As the triangulation process is not unique we can add the link B—E instead of I—D.

3.2.3 Junction tree

It is now possible to identify subgraphs, where all nodes are pairwise linked. Maximal subgraphs with this property are called cliques and are used as nodes afterwards in junction tree. The cliques of Fig.5 (C) are : {A,B,I}, {B,I,D}, {I,D,E},{D,E,F}, {I,G,E}, and {G,H,E}. To construct a junction tree the cliques are organized in a special way, so that all cliques C_i on a path $C_s - C_k - \ldots C_n - C_e$ between the start clique C_s and the end clique C_e contain the nodes of the intersection between C_s and C_e. Formally, $(C_s \cap C_e) \subseteq C_i, \forall C_i \in C_s - C_k - \ldots - C_n - C_e$. This property is known as the running intersection property (Cowell et al., 1999). According to (Jensen, 2001), there is always a way to organize the cliques of a triangulated graph into a junction tree. For inference purposes additional nodes containing the random variables in the intersection of two neighboured cliques are added (Cowell et al., 1999). These additional nodes are called separators (see Fig.5 (d)). The junction tree represented by Fig.5 (d) satisfies the running intersection property. For example we choose two cliques C_s = {A, B, I} and C_e= {I, G, E} with common nodes "I": {A, B, I} ∩ {I, G, E} = {I}. According to the definition of the running intersection property the variable "I" belong in all the cliques in the unique path between C_s and C_e. Effectively, all cliques and separators between the clique C_s= {A, B, I} and C_e= {I, G, E} contain the variable "I".



(a) Initial graph (b) Moralization (c) Triangulation (d) Junction tree

Fig. 5. Transformation steps from the initial Bayesian network to a junction tree

3.3 Initializing the junction tree

The junction tree should be initialized for use it in the inference of Bayesian network (computation of the posterior probabilities of some variables given observation). To enable the calculation of the distributions, tables are attached to each clique and separator of the junction tree, similar to the conditional probability tables of a Bayesian network. These tables are called potentials (Cowell et al., 1999), denoted by ψ_C and ψ_S, e.g. the potential of a clique C is denoted by ψ_C and potential of a separator S is denoted by ψ_S. Given the conditional probability distributions $P(X_i|\Pi_i)$ of the variables X_i (or $P(X_i)$ if there are no parents) the initialization of the junction tree can be performed as follows.

First, assign each variable X_i to just one clique that contains Π_i. The moralization process guarantees that a node X_i with all of its parent Π_i will occur together at least in a clique. For all instance of this variable affect 1. For example, at the clique {A, B, I} (see Fig.5 (d)) we assign the variables A, B and I. On the other hand, we assign just D at clique {B, I, D} because: the variable B was affected already to {A, B, I} and the other hand Π_B= {A} $\not\in$ {B, I, D}.

Second, for each clique C that is assigned with at least one variable define the potential function as the product of $P(X_i | \Pi_i)$ over all X_i assigned to C. For all separators and remaining cliques define the potential function to be 1. Table 1 summarizes the process of initializing of junction tree given by Fig.5 (d).

Cliques	Assigned variables	Potential cliques
{A, B, I}	A, B, I	$\psi_{\{A, B, I\}}$= P(A) . P(B | A) . P(I | A)
{B, I, D}	D	$\psi_{\{B, I, D\}}$= P(D | B)
{I, D, E}	E	$\psi_{\{I, D, E\}}$= P(E | I)
{D, E, F}	F	$\psi_{\{D, E, F\}}$= P(F | D, E)
{I, G, E}	G	$\psi_{\{I, G, E\}}$= P(G | I)
{G, H, E}	H	$\psi_{\{G, H, E\}}$= P(H | E, G)

Table 1. Cliques associated to initial directed acyclic graph (Fig.5 (a))

Associating a separator potential function to each separator set, we have the following factorization of the joint probability density (Cowell et al., 1999); (Murphy, 2002):

$$P(X_1,...,X_n) = \frac{\prod_{c\in C} \Psi_c(X_C)}{\prod_{s\in S} \Psi_s(X_S)} \tag{7}$$

3.3.1 Flow of information between adjacent cliques

The junction tree potential does indeed satisfy the equation (7), but it might not be consistent. Currently, comparing cliques they might tell you different stories about the distributions on certain variables. For instance, consider two adjacent cliques C_i and C_j with separator S, and both contains the variable "X". Marginalizing on both C_i and C_j to get "X", might not give the same result. The reason for this is that the individual cliques have not yet been affected by the information of other cliques in the tree. The information in the tree has to be distributed "equally" on. The global propagation algorithm performs a series of local manipulations that makes the junction tree locally consistent. To ensure consistency of the junction tree, messages are passed between the cliques of the junction tree. This results in a recalculation of the potentials. A clique C_j is said to absorb knowledge from a clique C_i, if the separator S between C_i and C_j gets as new potential Ψ_S^* (Cowell et al., 1999):

$$\Psi_S^*(X_S) = \sum_{C_i\setminus S} \Psi_{C_i}(X_{C_i}) \tag{8}$$

Namely, the separator potential $\Psi_S(X_S)$ is updated by marginalizing the clique potential over the variables that are in C_i but not in S. Then the update factor of the separator is used to update the potential of the destination clique C_j.

$$\lambda_S(X_S) = \frac{\Psi_S^*(X_S)}{\Psi_S(X_S)} \tag{9}$$

The new potential clique of C_j is given by:

$$\Psi^*_{C_j}(X_{C_j}) = \Psi_{C_j}(X_{C_j}).\lambda_S(X_S) \qquad (10)$$

In order to distribute the information in each clique to the whole tree a two-phase propagation algorithm is used. Given a root clique in the tree, the collection-phase absorbs the flows (messages) starting from the leafs towards the root. Once all the flows are collected in the root, messages are send towards the leafs in the distribution-phase. After collection-phase and distribution-phase are finished, it is guaranteed that the junction tree is globally consistent. Namely, for all clique C_i and C_j which are connected with a separator S, and both contains the variable "X", marginalizing on both C_i and C_j to get "X", might give the same result.

3.3.2 Entering and propagation evidence

The algorithm of initializing the junction tree can be used to compute posterior distribution given an observation of a subset of variables. Once the clique tree is properly initialized with the observation, the two phase propagation algorithm diffuses the observation and the resultant potentials are the posterior marginal distributions in each clique and separator. The observations are used in the algorithm as follows. Let X^h be the subset of variables that are hidden or unobserved and let D_l= {X_i=x_i, X_j=x_j,...} be an observation of the set of variables D_l =$X \backslash X^h$.. In order to compute $P(X^h | D_l)$ we first define an evidence function such that:

$$g(x_i) = \begin{cases} 1 & if\ x_i = x_i^* \\ 0 & otherwise \end{cases} \qquad (11)$$

After initializing the junction tree with the conditional probability distributions, we multiply each clique potential with the evidence function according to the variable assignment in the initialization step:

$$\Psi_C(X_C) = \Psi_C(X_C).\prod_{i:X_i \in X_C} g(x_i) \qquad (12)$$

Once again, we call collection-phase and distribution-phase to propagate this evidence (observation) through the tree to yield the new (posterior) probabilities. The clique potentials are the joint probability of the local hidden variables and the observed evidence:

$$\Psi_C(X_C) = P(X^h, D_l) \qquad (13)$$

If the potential function at a clique is normalized to sum to 1, one gets the conditional probability of the local hidden variables given the observation:

$$P(X^h \mid D_l) \qquad (14)$$

If the clique potential is marginalized over the hidden variables the probability of the observed evidence is obtained:

$$P(D_l) = \sum_{X^h} P(X^h, D_l) \qquad (15)$$

3.4 Bayesian network model for Mono-vehicle localization

Vehicle localization on respect of a road map involves applying a first filter which selects all the segments close to the estimated position of the vehicle. The goal is then to select the most likely segment(s) from this subset. Nowadays, since the geometry of roadmaps is more and more detailed, the number of segments representing roads is increasing. On the other hand, the map is not perfect and presents a known uncertainty. The road classification module is an important stage in the vehicle localization process because the robustness of the localization depends mainly on this stage. In order to take into account the error of several sensors or database used in this application, we introduce a concept which can manage multi-hypothesis in the formalism of Bayesian network.

For each selected segment $Carto_i$, we represent it by a Gaussian: $Carto_i \sim N(\mu_i, \Sigma_i)$. Where $\mu_i=(x_i,y_i,\theta_i)$ is the projection of the estimated position on this segment and θ_i is the heading of the segment. The proposed Bayesian network model for Mono-vehicle localization is illustrated in Fig. 6. In this model we used two hidden variables. The discrete variable Sk represents the segments of which the vehicle can be. The second is continuous variable; $X_k(x_k,y_k,\theta_k)$ represents the estimation of a vehicle for each candidate segment. The graph represented in Fig. 6. allows us to represent causal links between the variables. The continuous variable X_k is updated by observations $Carto_k$ and GPS_k if GPS measurement is available. This variable is multi-modal because it has been updated by the set of candidates segments ($Carto_k$). The discrete variable S_k is update by cartographical observation ($Carto_k$) and the estimation is given by the hidden variable X_k.

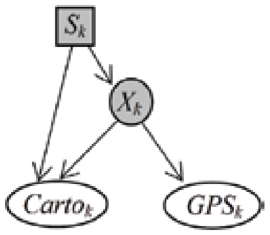
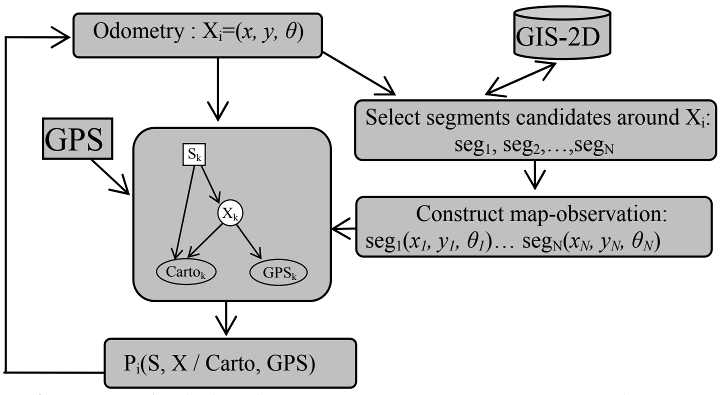


Fig. 6. Bayesian network model for Mono-vehicle localization

Bayesian inference gives us :

1. the probability of each candidate segment given by the posterior probabilité $P(S_k \mid Carto_k,GPS_k)$
2. for each segment, the estimation of the vehicle's position given by the probability density $P(X_k \mid Carto_k,GPS_k)$



Odometry : $X_i=(x, y, \theta)$
GIS-2D
GPS
Select segments candidates around X_i: $seg_1, seg_2,\dots,seg_N$
Construct map-observation: $seg_1(x_1, y_1, \theta_1)\dots seg_N(x_N, y_N, \theta_N)$
S_k
X_k
$Carto_k$
GPS_k
$P_i(S, X / Carto, GPS)$

Fig. 7. Synoptic of Mono-vehicle localization using a Bayesian network

Let us use a specific case study to illustrate the method. In Fig. 8, the vehicle is travelling from the road 1 to road 8. At t=1, we have a predicted pose given by the odometer. We select all segments around the predicted pose. In this case we have one. This segment was used to generate one cartographical observation. Then, estimation is provided using Bayesian network. At t=2, the estimation errors and the digital map errors oblige to select segments 2 and 3 around the predicted pose. These segments were used to generate two observations. Then, using Bayesian network, two estimations were provided and so on, for the rest of the experiment. Table 2 summarizes the process of inference Bayesian.

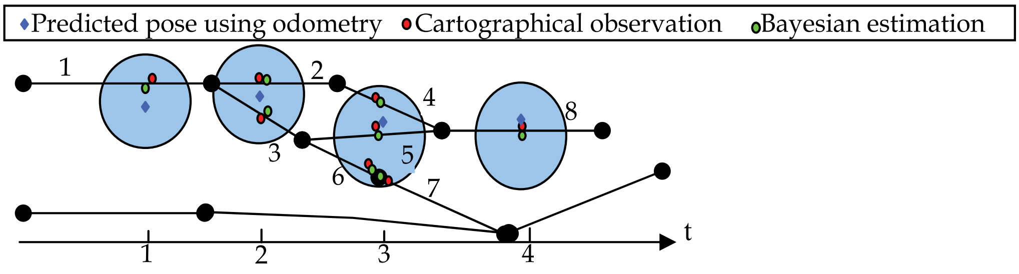


Fig. 8. Formalism of multi-hypothesis managing in Bayesian network

t	Carto	$P(S_t \mid Carto)$	$P(X \mid Carto)$
1	{1}	$P(S_t=\{1\} \mid \{1\})=1$	$X_{\{1\}}=(x_{\{1\}}, y_{\{1\}}, \theta_{\{1\}})$
2	{2,3}	$P(S_t=\{2\} \mid \{2,3\})=0.75$ $P(S_t=\{3\} \mid \{2,3\})=0.25$	$X_{\{2\}}=(x_{\{2\}}, y_{\{2\}}, \theta_{\{2\}})$ $X_{\{3\}}=(x_{\{3\}}, y_{\{3\}}, \theta_{\{3\}})$
3	{4,5,6,7}	$P(S_t=\{4\} \mid \{4,5,6,7\})=0.55$ $P(S_t=\{5\} \mid \{4,5,6,7\})=0.25$ $P(S_t=\{6\} \mid \{4,5,6,7\})=0.15$ $P(S_t=\{7\} \mid \{4,5,6,7\})=0.05$	$X_{\{4\}}=(x_{\{4\}}, y_{\{4\}}, \theta_{\{4\}})$ $X_{\{5\}}=(x_{\{5\}}, y_{\{5\}}, \theta_{\{5\}})$ $X_{\{6\}}=(x_{\{6\}}, y_{\{6\}}, \theta_{\{6\}})$ $X_{\{7\}}=(x_{\{7\}}, y_{\{7\}}, \theta_{\{7\}})$
4	{8}	$P(S_t=\{8\} \mid \{8\})=1$	$X_{\{8\}}=(x_{\{8\}}, y_{\{8\}}, \theta_{\{8\}})$

Table 2. Example of inference process given by Bayesian network

4. Bayesian network model for multi-vehicle localization

In order to move a set of vehicles in train configuration, each vehicle must to know its position. In our approach, we propose that the path of the leader vehicle is propagated in real time to follower's vehicle using wireless communication. Each vehicle in the platoon is equipped with: GPS sensors in order to determine its location and a Lidar to calculate the distance to vehicle in front. Only the leader vehicle is equipped with RTK-GPS and a special localization device. This device should provide geo-position with high accuracy.

Each follower vehicle can transmit its position and velocity. To reproduce the path of the leader vehicle, each vehicle constructs the trajectory of the leader vehicle by linking up the position transmitted by the leader. Then the follower can calculate the lateral and longitudinal controls to be close to the leader vehicle path and to keep a constant distance within vehicles. Follower's vehicles positioning is improved by using the Lidar and propagated measure of leader vehicle. According to Fig.9, the Cartesian coordinates of first follower (f_1) are given by:

$$X_{Vobs}^{f_1} = \begin{cases} X_{\text{Leader}} - D.\cos(\beta) + N(\mu,\sigma) \\ Y_{\text{Leader}} - D.\sin(\beta) + N(\mu,\sigma) \end{cases} \tag{16}$$

D is the distance (given by rangefinder) between the follower and the leader. The same process is carrying out between follower (f_{i+1}) and follower (f_i):

$$X_{Vobs}^{f_i} = \begin{cases} X_{f_{i-1}} - D.\cos(\beta) + N(\mu,\sigma) \\ Y_{f_{i-1}} - D.\sin(\beta) + N(\mu,\sigma) \end{cases} \tag{17}$$

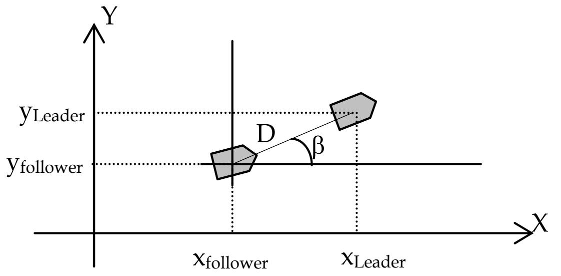


Fig.9. Approving follower's position according to leader's position and rangefinder data

The proposed Bayesian network model for multi-vehicle localization is illustrated in Fig.10. In this model we used the continuous hidden variable $X_{S_i}(k+1) = (x_{S_i}^{k+1}, y_{S_i}^{k+1}, \theta_{S_i}^{k+1})$ for each follower vehicle to estimate its position. This variable is updated by the observations $X_{gps\_Si}(k+1)$ and $X_{Vobs\_Si}(k+1)$ and depends on the precedent state $X_{Si}(k)$ and law command $U_{Si}(k)$. Finally, the law command (lateral and longitudinal) depends on the trajectory of leader the vehicle $X_L(k+1) = (x_L^{k+1}, y_L^{k+1}, \theta_L^{k+1})$.

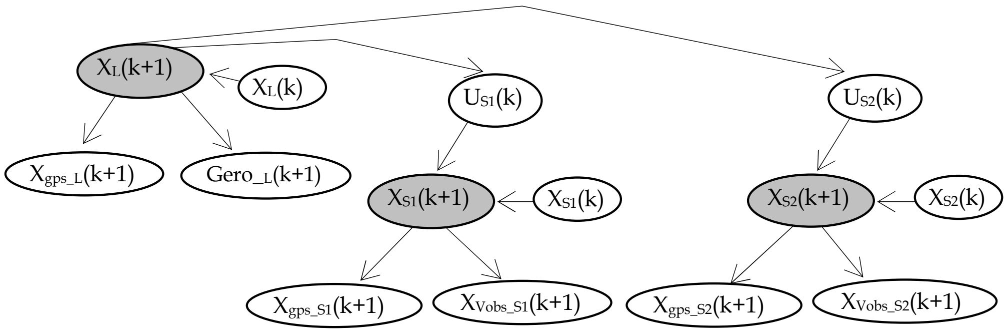


Fig.10. Bayesian network model for Multi-vehicle localization

Each vehicle has to know its position to derive an adequate control law which allows it to:

- preserve a constant distance (DistS) to the vehicle in front
- reduce lateral distance (DistL) between the trajectory of the follower and the leader one (Fig.11).

Each vehicle's position is represented by the (x_k,y_k) Cartesian coordinates of M. The heading angle is denoted θ_k. The motion model can be expressed as:

$$X_{k+1} = \begin{cases} x_{k+1} = x_k + d_s.\cos(\theta_k + \frac{\omega_\theta}{2}) \\ y_{k+1} = y_k + d_s.\sin(\theta_k + \frac{\omega_\theta}{2}) \\ \theta_{k+1} = \theta_k + \omega_\theta \end{cases} \tag{18}$$

Where d_s is the length of the circular arc followed by M and ω_θ is the elementary rotation of the mobile frame.

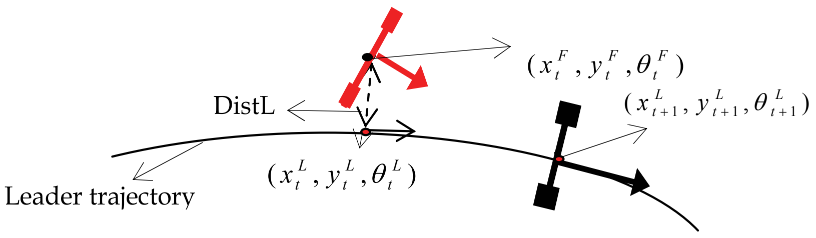

$(x_t^F, y_t^F, \theta_t^F)$
DistL
$(x_{t+1}^L, y_{t+1}^L, \theta_{t+1}^L)$
Leader trajectory
$(x_t^L, y_t^L, \theta_t^L)$

Fig.11. Model for leader vehicle with one follower

5. Experiment results

A test trajectory for a mono-vehicle localization has been carried out at Compiègne in France with an experimental vehicle. The used GPS is a differential Trimble AgGPS132 receiver. For odometry, we have used the ABS sensors of the rear wheels of the experimental vehicle.
For multi-vehicle localization, a test trajectory was carried out at the place Stanislas at Nancy-France for the leader vehicle. For followers' vehicle GPS and Lidar are simulated by adding Gaussian noise for followers' vehicles.

5.1 Performance of mono vehicle localization method

The test trajectory is presented in Fig.12. In this experience, the GPS measurements were available in the beginning of the test trajectory. Then, the GPS was not used for 1.5Km. One can remark that in spite of the long GPS mask, the vehicle location is matched correctly. As a matter of fact, the final estimated positions stay close to the GPS points. In Fig.12, we only presented the most probable Bayesian network estimation of the pose.
In second experiment (see Fig. 13), we want to show how the Bayesian network handles and treats ambiguous situations (junction road and parallel road). To simulate satellite masks occurring in urban environments we remove GPS data and put them back again. In the first situation, GPS was not available before the junction. One can see that the method manages two hypotheses (two segments) for three steps then wrong hypothesis was eliminated by presence the GPS. One can remark that among the different hypotheses the good segment was always given the highest probability by the Bayesian network inference. On the same figure, a second ambiguous situation appears: close and parallel road. The method detects the ambiguity of this situation and selects all probable segments in this parallel roads. The Bayesian network manages all hypotheses until the elimination of the ambiguity. One can remark that in spite the ambiguity the road on which the vehicle is running presents the highest probability.

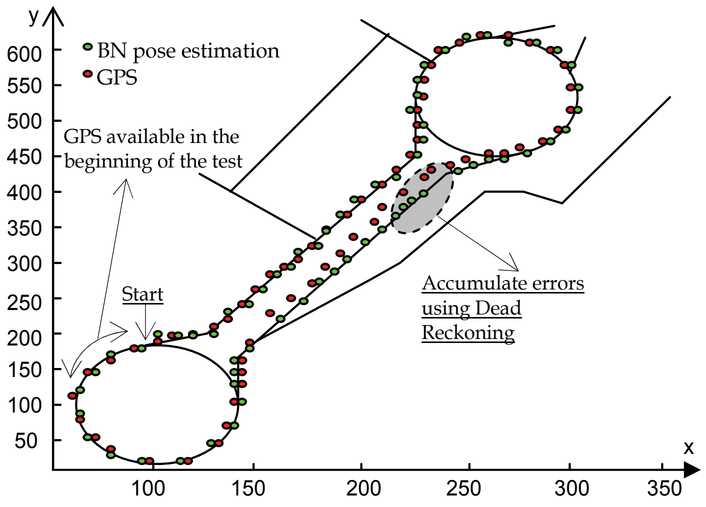


Fig.12. Pose estimation with Bayesian network using odometry and cartographical observation (GPS was masked)

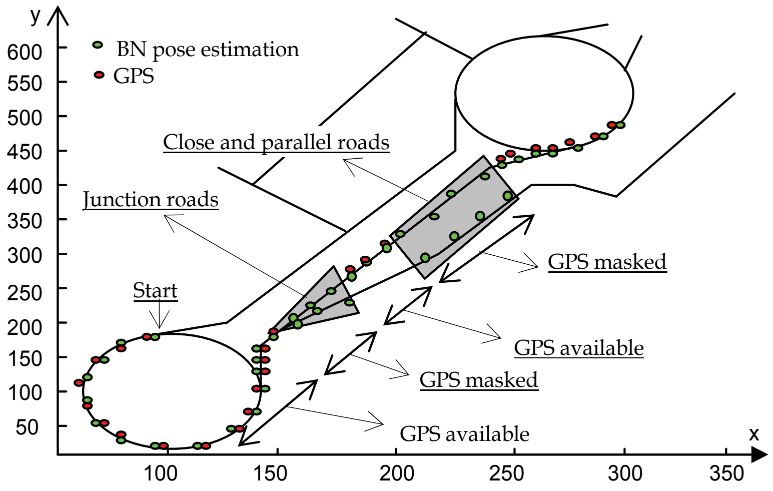


Fig.13. Multi-hypothesis managed with Bayesian network to treat junction road and parallel road situation

5.2 Performance of multi-vehicle localization

The real trajectory of the vehicle leader is presented by the green line on Fig.14. This trajectory represents the centimetric precision GPS position. These geo-positions are provided by the THALES Sagitta02 which is a centimetric GPS LRK. In this experiment, we assume that all of two follows vehicles (blue and black respectively) are initially parked on the reference path and not necessarily have the same leader's heading. Followers dispose by WiFi and in real time the path of the leader vehicle.

The leader's path, given by centimetric GPS-LRK, is plotted by green line in Fig.14. The figure 15 shows the time evolution of the vehicles positions provided by the proposed approach. Thus followers' vehicles are supposed to be equipped by low cost GPS sensors (3 meters accuracy) for localization task and a rangefinder to determine distance between vehicles. The obtained followers' vehicles paths are plotted in Fig.15 by blue path for first follower and red path for second one.

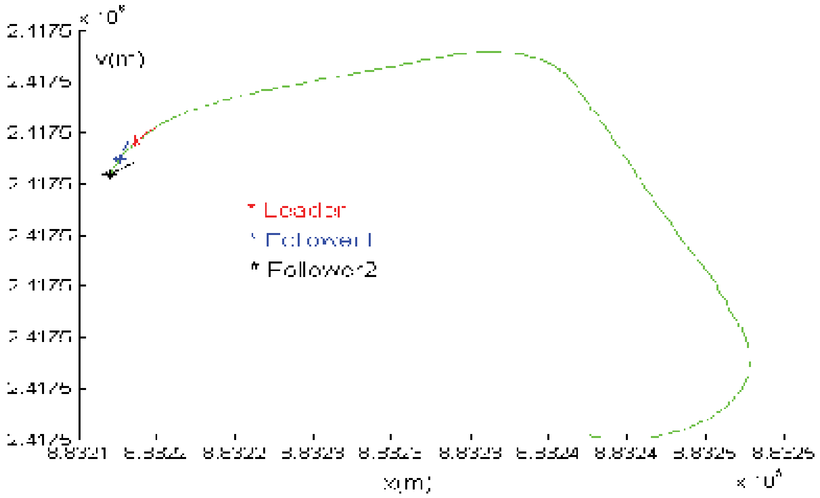


Fig.14. Leader's trajectory and initial position of vehicles

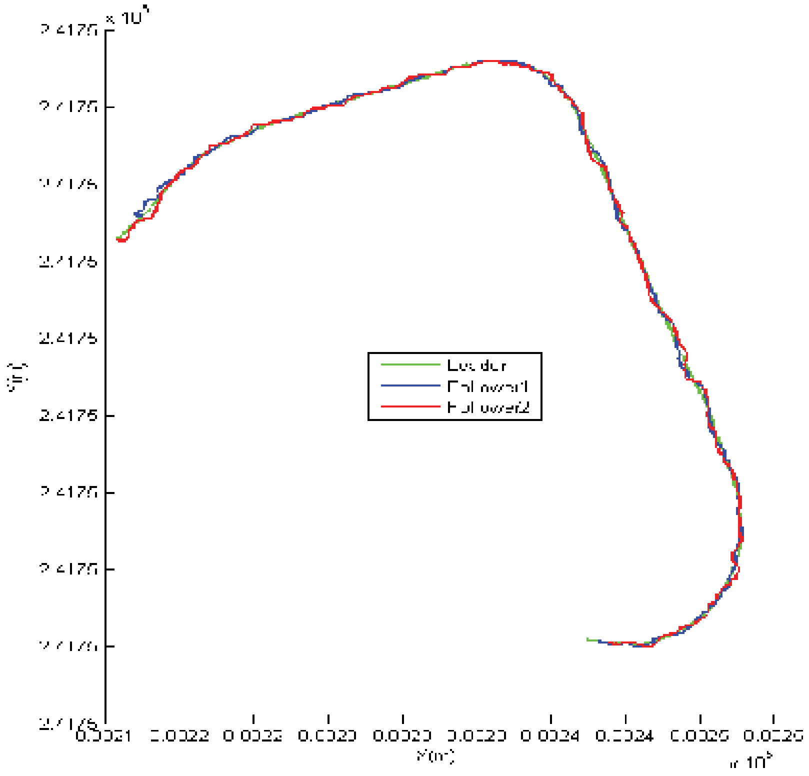


Fig.15. Followers' trajectory given by Bayesian network compared to leader's trajectory

One can remark that the followers' paths shown in figure 15 present oscillations all around leader path. We believe that oscillations come from the used control law. In this work, a simple proportional law control is used.
Figure 16 show the distance between vehicles. Initially, inter-distance between vehicles is chosen to be for about 2 meters. In this simulation, the desired security distance between vehicles is chosen to be 1 meter. Namely, the objective is to control vehicles in order to respect this distance. It can be noticed that the chosen control law in each vehicles commands the vehicle velocity in order to respect the desired security distance. One can remark that the security distance is generally respected.

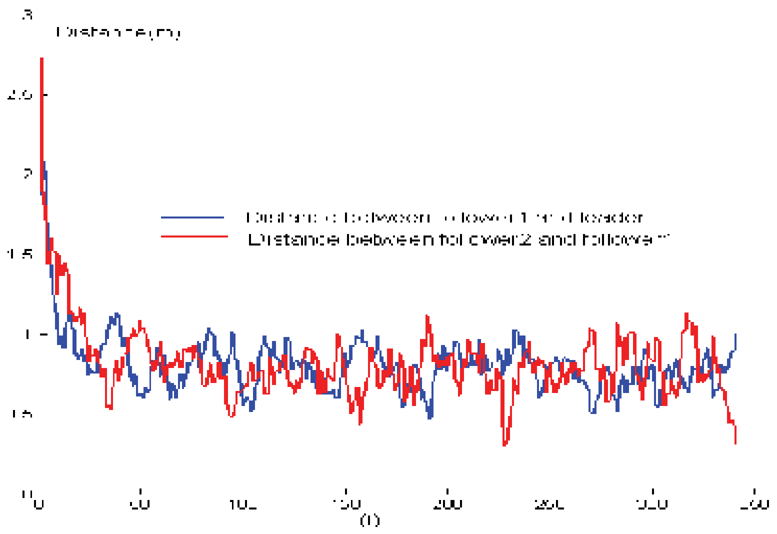


Fig.16. Distance between vehicles

7. Conclusion

This article has presented a multi-sensor fusion method for vehicle localization. The main contributions of this work are the formalization of a multi-sensor fusion method in the Bayesian Network context and an experimental validation with real data.
An interesting characteristic of this approach is that it is flexible and modular in the sense that it can easily integrate other sensors. This feature is interesting because adding other sensors is a way to increase the robustness of the localization.
In this approach, the use of the digital map as an observation of the state space representation has been introduced. This observation is used in the Bayesian Network framework in the same way that the GPS measurement. It turned out in the experiments that the GPS measurements are not necessary all the time, since the merging of odometry and roadmap data can provide a good estimation of the position over a substantial period.
The strategy presented in this paper doesn't keep only the most likely segment. When approaching an intersection, several roads can be good candidates for this reason we manage several hypotheses until the situation becomes unambiguous.
Multi-vehicle localization method presented in this work can be seen like an extension of the Mono-vehicle method, in the sense that we have duplicate the (BN) used to fuse

measurements sensors to localize one vehicle for several ones. Then, we have added vehicles inter-connection to represent finally the train of vehicles in the context of Bayesian network. The multi-sensor fusion of leader's vehicle measurements with the Lidar measurement in Bayesian network formalism can provide continuous and accurate geo-position of followers' vehicles. Thus this data fusion method allows computing an accurate follower's position without using an expensive sensor on each follower's vehicle.
The proposed method for multi-sensors fusion for multi-vehicle localization in train configuration permits to implement control law based on near-to-near approach, which can only be seen as a first step in platoon control design. For both proposed approach, real data was used in this work to test and quantify the quality of results.

8. Future work

In this work, we have not treated the control law of the follower vehicles. The perspective of this work is to use the tricycle model validated by numerous laboratories (Daviet & Parent); (Thuilot et al., 2004). The tricycle model allows us to manipulate the curvilinear distance instead of rectilinear distance. The main advantage of curvilinear distance is that it agrees with the distance travelled and is perfectly consistent when following reference paths with high curvature (which is not the case with rectilinear distance) (Bom et al., 2005). Using curvilinear distance allows us to reduce the difference between follower's trajectory and leader's one. Moreover, platoon control design relies on nonlinear techniques, instead of control approaches based on linear approximations. Since no approximation is achieved, performances provided by the nonlinear control law are more satisfactory and more robust than those offered by linear control. This control approach allows to fully decouple longitudinal and lateral controls (Bom et al., 2005).

9. References

Abbott, E. & Powell, D. (1999). Land-Vehicle Navigation using GPS. *Proceedings of IEEE, vol.87, N.1*

Bom, J.; Thuilot, B.; Bom, J.; Marmoiton, F. & Martinet, P. (2005). Nonlinear Control for Urban Vehicles Platooning, Relying upon a Unique Kinematic GPS. *Proceedings of the 2005 IEEE International Conference*

Castillo, E.; José M. G. & Ali, S. H. (1997). *Expert Systems and Probabilistic Network Models,* Publisher, ISBN, New York Berlin Heidelberg

Cowell, R. G.; Dawid, A. P.; Steffen, L.; & David, J. S. (1999). *Probabilistic Network and Expert Systems,* Publisher, ISBN, New York Berlin Heidelberg

Daviet, P. & Parent, M. (1995). Platooning for small public urban vehicles, *In 4t hIntern. Symposium Experimental Robotics, pp 345-354*17, Stanford, CA (USA), July

Daviet, P. & Parent, M. (1996). Longitudinal and Lateral Servoing of Vehicles in a Platoon. *In Intelligent Vehicles Symposium*. Tokyo Japan

Dissanayake, M.W.; Newman, P.; Clark, S.; Durrant-Whyte, H.F.; & Csorba, M. (2001). A Solution to the Simultaneous Locakization and Map Building (SLAM) Problem. IEEE Transaction on Robotics and Automation, Vol. 17, N0.3

EL Najjar, M. E. & Bonnifait, P. (2003). A Road Matching Method for Precise Vehicle Localization using Belief Theory and Kalman Filtering. *IEEE/EJS/ISR 11th Int. Conference on Advanced Robotics, pp.1677-1682*

EL Najjar, M. E. & Bonnifait, Ph. (2005). To wards an Estimate of Confidence In a Road-Matched Location. IEEE International Conference on Robotics and Automation

Heckerman, D. (1995). A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Advanced Technology Division

Judea, P. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference,* Publisher, ISBN, San Francisco

Jensen, F. V.; Lauritzen, S. L.; Steffen, & Olsen, K. G. (1990). *Bayesian updating in recursive graphical models by local computations,* Computational Statistics and Data Analysis Publisher, ISBN

Jensen, F. V. (2001). *Bayesian networks and Decision Graphs,* Statistics for Engineering and Information Science. Springer, Berlin, Heidelberg

Kjaerulff, U. (1990). Triangulation of graphs-algorithms giving small total space. Department of Mathematics and Computer Science. Institute of Electronic Systems. Aalborg University

Murat Deviren (2001). "Structural Learning of Dynamic Bayesian Networks in Speech Recognition". Technical Report.

Murphy, K. P. (2002). Dynamic Bayesian Networks: Representation, Inference and Learning. PhD thesis, UC Berkley, Computer Science Division

Thrun, S.; Fox, D.; Burgard, W. & Dellaert, F. (2001). Robust Monte-Carlo localization for mobile robots. *Journal of Artificial Intelligence (AI). Vol. 128, No 1-2. pp 99-141*

Thuilot, B.; Bom, J.; Marmoiton, F. & Martinet, P. (2004). Accurate automatic guidance of an urban electric vehicle relying on kenematic GPS sensor. *in 5th IFAC Symposium on Intelligent Autonomous Vehicles (IVAC'04),* July

Yannakakis, M. (1981). Computing the minimal fill-in in np-complete. *SIAM Journal on Matrix Analysis and Applications,* Vol. 2, Issue 1, page 77-79

Zhao, Y. (1997). *Vehicle Location Navigation Systems,* Artech House Publishers

On the Definition of a Standard Language for Modelling Constraint Satisfaction Problems

Ricardo Soto[1,2], Laurent Granvilliers[1]
[1]*CNRS, LINA, Université de Nantes*
[2]*Escuela de Ingeniería Informática,*
Pontificia Universidad Católica de Valparaíso,
Chile

1. Introduction

A Constraint Satisfaction Problem (CSP) is a declarative representation of a system under constraints. Such a representation is mainly composed by two parts: a sequence of variables lying in a domain and a finite set of constraints over these variables. The goal is to find values for the variables in order to satisfy the whole set of constraints.

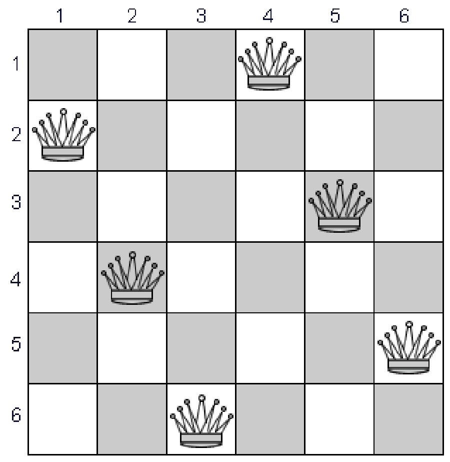


Fig. 1. A solution of the 6-queens problem.

As an example, let us consider the 6-queens problem, which consists in placing 6 chess queens on a 6x6 chessboard such that none of them is able to capture any other using the standard chess queen's moves. A solution requires that no two queens share the same row, column, or diagonal. A CSP for this problem can be stated by means of six variables and three constraints. One variable for each of the six queens to represent their row positions on the chessboard, each variable lying in the domain [1,6] (6 rows). One constraint to avoid that two queens are placed in the same row. One constraint to avoid that two queens are placed in the first diagonal; and finally one constraint to avoid that two queens are placed in the second diagonal. A solution of the 6-queens problem is shown in Figure 1.

Constraint Programming (CP) is called the software technology to solve CSPs. Currently, several CP tools exist which are mainly systems or libraries built on top of a host language such as Prolog (ECLiPSe (Wallace et al., 1997)), C++ (ILOG Solver (Puget, 1994)) or Java (Gecode/J (Schulte & Stuckey, 2004)). These tools, generically named solvers, are able to take as input the CSP and to solve it by exploring and pruning efficiently the search space containing the potential solutions. A main advantage of this approach is that users just need to state a declarative representation of the problem, instead of building complex procedures or algorithms to search the solutions. This is a great asset; however, the language to express these problems is not standard, each tool provides its own semantics with a level of abstraction tied to its host language, which is commonly not easy to use. This is a big concern in the CP field, since users need to deal with the encoding concerns of the solver host language; and moreover to learn a new language each time they want to experiment with a new solver.

In response to this, the CP community has defined the development of a standard language as an important research direction. To this end, a solver-independent three layered architecture has been proposed (Rafeh et al., 2007; Fritsh et al., 2007), including a modelling language -which is expected to be simple enough for use by non CP experts-, a set of solvers and a middle tool mapping models to executable solver code. Thus, the user is able to design one model in a "human-comprehensible" language and to target many solvers.

In this work, we follow this research direction by proposing a solver-independent modelling language but using an object-oriented approach. Our system is called s-COMMA (Soto & Granvilliers, 2007) and it can be regarded as a hybrid built from a combination of an object-oriented language and a constraint language. The s-COMMA constraint language provides typical data structures, control operations, and first-order logic to define constraint-based formulas. The object-oriented part is a simplification of the Java programming style. This framework clearly provides model structures using composition relationships.

The s-COMMA system is written in Java (22000 lines) and it is supported by a solver-independent execution platform where models can be solved by four well-known CP solvers: Gecode/J, ECLiPSe, RealPaver (Granvilliers & Benhamou, 2006) and GNU Prolog (Diaz & Codognet, 2000). We believe s-COMMA is in compliance with the requirements of a standard language. Their simplicity is similar to the state-of-the-art modelling languages (Nethercote et al., 2007; Fritsh et al., 2007). The expressiveness provided is considerable and even it can be increased with extension mechanisms. The solver-independence is the base of the platform which allows experimentations with many solvers.

The definition of a standard language is an evident hard task which may require many years and several experimental steps. We believe that the work done on s-COMMA is one of the steps towards the achievement of this important goal.

The outline of this chapter is as follows. Section 2 introduces an overview of the s-COMMA language. The architecture and implementation of the system is explained in Section 3. The related work is presented in Section 4 followed by the conclusions.

2. s-COMMA overview

In this section we give an overview of s-COMMA. We will first present the most important elements of an s-COMMA model and then, we will illustrate these elements by means of three examples, the n-queens problem, the packing squares problem and a production-optimization problem.

2.1 s-COMMA models

An s-COMMA model is composed by two main parts, a model file and a data file. The model file describes the structure of the problem, and the data file contains the constant values used by the model. The model file is composed by import statements and classes; and the data file is composed by constants and variable assignments.

2.1.1 Constants & variable assignments

Constants, namely data variables, are stated in a separate data file and imported from the model file. Constants can be real, integer or enumeration types. Arrays of one dimension and arrays of two dimensions of data variables are allowed. A variable-assignment is an assignment of a value to a variable of an object defined in the model file (as example, see line 3 of the data file in Fig. 4).

2.1.2 Classes

A class is composed by attributes and constraints zones. Single inheritance is permitted and a subclass inherits all attributes and constraints of its superclass.

2.1.3 Attributes

Attributes may represent decision variables or objects. Decision variables must be declared with an integer, real or boolean type. Objects are instances of classes which must be typed with their class name. Arrays of one and two dimensions can be used; they can contain either decision variables or objects. Decision variables and arrays of decision variables can be constrained to a determined domain.

2.1.3 Constraint zones

Constraint zones are used to group constraints encapsulating them inside a class. A constraint zone is stated with a name and it can contain constraints, forall loops, if-else statements, optimization statements, and global constraints.

2.2 The n-queens problem

Let us begin the illustration of these elements by means of the n-queens problem presented in Section 1. An s-COMMA model for this problem is shown in Figure 2. The model is represented by a class called `Queens` which contains an array with `n` integer decision variables lying in the domain `[1,n]`. The constant value called `n` is imported from the `Queens.dat` file.

At line 6 a constraint zone called `noAttack` contains the three constraints required. One constraint to avoid that two queens are placed in the same row (line 9). One constraint to avoid that two queens are placed in the first diagonal (line 10); and one constraint to avoid that two queens are placed in the second diagonal (line 11). Two for loops ensure that the constraints are applied for the complete set of decision variables.

2.3 The packing squares problem

Let us continue with a more complex problem called packing squares. The goal of this problem is to place a given set of squares in a square area. Squares may have different sizes and they must be placed in the square area without overlapping each other. Figure 3 shows a solution for the problem, 8 squares have been placed in a square area of side size 5.

```
//Data file
n:=6;

//Model file
import Queens.dat;

class Queens {
  int q[n] in [1,n];
  constraint noAttack {
    forall(i in 1..n) {
      forall(j in i+1..n) {
        q[i] <> q[j];
        q[i]+i <> q[j]+j;
        q[i]-i <> q[j]-j;
      }
    }
  }
}
```

Fig. 2. s-COMMA model for the n-queens problem.

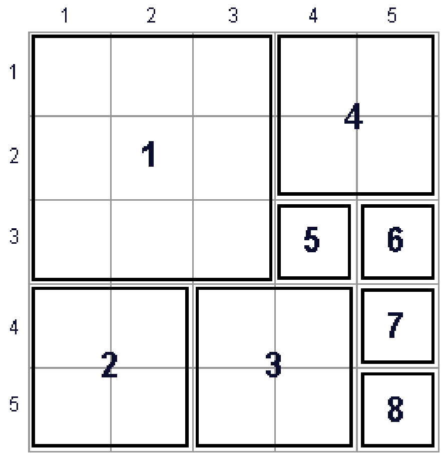


Fig. 3. A solution of the packing squares problem.

Figure 4 shows an s-COMMA model for the packing squares problem. Data values are imported from an external data file called `PackingSquares.dat`, `sideSize` (line 1) is an integer constant that represents the side size of the square area where squares must be placed, `squares` (line 2) represents the quantity of squares to place. `PackingSquares.s` (line 3) is a variable-assignment for the array of `Square` objects declared at line 5 of the model file, here a set of values is assigned to the third attribute (`size`) of each `Square` object of the array `s`. For instance, the value `3` is assigned to the attribute `size` of the first object of the array. The value `2` is assigned to the attribute `size` of the second, third and fourth object of the array. The value `1` is assigned to the attribute `size` of remaining objects of the array. We use standard modelling notation ('_') to omit assignments.

At line 3 in the model file, the definition of the class begins, `PackingSquares` is the name given to this class. Then, an array containing objects from the class `Square` is defined. This class, declared at line 30, is used to model the set of squares. Attributes `x` and `y` represent respectively the x and y coordinates where the squares must be placed. So, `s[8].x=5` and `s[8].y=5` means that the eighth square must be placed in row 5 and column 5, indeed in the bottom right corner of the square area. Both variables (x,y) are constrained, they must have values into the domain `[1,sideSize]`. The last attribute called `size` represents the size of the square.

```
//Data file
1.  int sideSize :=5;
2.  int squares :=8;
3.  Square PerfectSquares.s := [{_,_,3},{_,_,2},{_,_,2},{_,_,2},
                                {_,_,1},{_,_,1},{_,_,1},{_,_,1}];


//Model file
1.  import PackingSquares.dat;
2.
3.  class PackingSquares {
4.
5.    Square s[squares];
6.
7.    constraint inside {
8.      forall(i in 1..squares) {
9.        s[i].x <= sideSize - s[i].size + 1;
10.       s[i].y <= sideSize - s[i].size + 1;
11.     }
12.   }
13.
14.   constraint noOverlap {
15.     forall(i in 1..squares) {
16.       forall(j in i+1..squares) {
17.         s[i].x + s[i].size <= s[j].x or
18.         s[j].x + s[j].size <= s[i].x or
19.         s[i].y + s[i].size <= s[j].y or
20.         s[j].y + s[j].size <= s[i].y;
21.       }
22.     }
23.   }
24.
25.   constraint fitArea {
26.     sum(i in 1..squares)(s[i].size*s[i].size)) = sideSize*sideSize;
27.   }
28. }
29.
30. class Square {
31.   int x in [1,sideSize];
32.   int y in [1,sideSize];
33.   int size;
34. }
```

Fig. 4. s-COMMA model for the packing squares problem.

At line 7, a constraint zone called `inside` is declared. In this zone a `forall` loop contains two constraints to ensure that each square is placed inside the area, one constraint about rows and the other about columns. Let us note that loops use loop-variables which do not need to be declared (`i` and `j` in the example).

The constraint zone `noOverlap,` declared at line 14, ensures that two squares do not overlap. The last constraint zone called `fitArea` ensures that the set of squares fits perfectly in the square area.

2.4 The production problem

Let us finish the s-COMMA overview with a production-optimization problem. Consider a factory that must satisfy a determined demand of products. These products can be either manufactured inside the factory -considering a limited resource availability- or purchased from an external market. The goal is to determine the quantity of products that must be produced inside the factory and the quantity to be purchased in order to minimize the total cost.
Figure 5 shows an s-COMMA model for this problem. At Line 28 of the model, the class to represent products is stated. Each `Product` is composed by its demand, its inside and outside cost, its consumption, and the quantity that must be produced inside and outside the factory. At line 3 the main class of the problem begins, it is first composed by two arrays, one containing the set of products and the other contains the available quantity of resources for manufacturing the products.
At Line 9 a constraint zone called `noExceedCapacity` is stated to ensure that the resource consumption of products manufactured inside do not exceed the total quantity of available resources. At line 15, `satisfyDemand` is posted to satisfy the demand of all the products. Finally, at line 22, an optimization statement is posted to determine the quantity of products that must be produced inside the factory and the quantity to be purchased in order to minimize the total cost.
The data file is composed by two enumerations that define the resources and the name's products respectively. At line 3, a variable-assignment for the `capacity` attribute of the class `Production` is stated. At the end, `Production.products` is a variable-assignment for the array `products` defined at line 5 of the model file. This variable-assignment states that the demand of the product klusky is 1000, their inside and outside cost are 6 and 8, respectively; and finally its production requires 5 flour items and 2 eggs. The assignment of the following products is analogous.

```
//Data file
1.  enum resourceList := {flour, eggs};
2.  enum productList  := {klusky, capellini, fettucine};
3.  int Production.capacity := [200,400];
4.  Product Production.products :=
         [klusky:{1000,6,8,[flour:5,eggs:2],_,_},

       capellini:{2000,2,9,[flour:4,eggs:4],_,_},

       fettucine:{3000,3,4,[flour:3,eggs:6],_,_}];


//Model File
1.  import Production.dat;
2.
3.  class Production {
4.
5.    Product products[productList];
6.    int capacity[resources];
```

```

  constraint noExceedCapacity {
    forall(r in resourceList) {
      capacity[r] >= sum(p in productList)
        (products[p].consumption[r] * products[p].inside);
    }
  }

  constraint satisfyDemand {
    forall(p in productList) {
      products[p].inside + products[p].outside >= products[p].demand;
    }
  }

  constraint minimizeCost {
    [minimize] sum(p in productList)
         (products[p].insideCost  * products[p].inside +
          products[p].outsideCost * products[p].outside);
  }
}

class Product {
  int demand;
  int insideCost;
  int outsideCost;
  int consumption[resourceList];
  int inside in [0,5000];
  int outside in [0,5000];
}
```

Fig. 5. s-COMMA model for the production problem.

2.5 Extension mechanism

Extensibility is an important feature of s-COMMA. Let us now show this feature using the packing squares problem. Consider that a programmer adds to the Gecode/J solver two new built-in functionalities: a constraint called `inside` and a function called `pow`. The constraint `inside` ensures that a square is placed inside a given area, and `pow(x,y)` calculates the value of `x` to the power of `y`. In order to use these functionalities we can use these new built-ins from s-COMMA by defining an extension file where the rules of the translation are described. This file is composed by one or more main blocks (see Figure 6). A main block defines the solver where the new functionalities will be defined. Inside a main block two new blocks are defined: a `Function` block and a `Relation` block. In the `Function` block we define the new functions to add. The grammar of the rule is as follows:

$$\langle name\rangle(\langle input - parameters\rangle) \to "\langle solver - code\rangle";$$

In the example, the left part of the rule is `pow(x,y)`, `pow` is the name of the function and `x` and `y` the input parameters. The left part of the rule corresponds to the statement that will be used to call the new function from s-COMMA. The right part corresponds to the code that calls the new built-in method from the solver file. Thus, the code `pow(x,y)` will be translated to `power(x,y)` from s-COMMA to Gecode/J. The translator must recognize the correspondence between input parameters in s-COMMA and input parameters in the solver code. Therefore, variables are tagged with '$' symbols. In the example, the first parameter

and the second parameter of the s-COMMA function will be translated as the first parameter and the second parameter in the Gecode/J function, respectively.
Within the `Relation` block we define the new constraints to add. In the example, a new constraint called `inside` is defined, it receives four parameters. The translation to Gecode/J is given in the same way. Once the extension file is completed, it can be called by means of an import statement. The resultant s-COMMA model using extensions is shown in Figure 6.

```
//Extension File
GecodeJ {
  Function {
    pow(x,y) -> "power($x$,$y$)";
  }
  Relation {
    inside(a,b,c,d) -> "inside($a$,$b$,$c$,$d$);";
  }
}

ECLiPSe {
  Function {
 ...
GNUProlog {
  Function {
 ...
RealPaver {
  Function {

//Model file
import PackingSquares.dat;
import PackingSquares.ext;

class PackingSquares {

   Square s[squares];

   constraint placeSquares {
     forall(i in 1..squares) {
       inside(s[i].x,s[i].y,s[i].size,sideSize);
       forall(j in i+1..squares) {
         s[i].x + s[i].size <= s[j].x or
         s[j].x + s[j].size <= s[i].x or
         s[i].y + s[i].size <= s[j].y or
         s[j].y + s[j].size <= s[i].y;
       }
     }
   }

   constraint fitArea {
        (sum(i in 1..squares) (pow(s[i].size,2)) = pow(sideSize,2);
   }
}
```

Fig. 6. s-COMMA model for the packing squares problem using extensions.

3. s-COMMA architecture

The s-COMMA system is supported by a three-layered architecture: Modelling, Mapping and Solving (see Fig 7). On the first layer, models are stated; extension and data files can be given optionally. Models are syntactically and semantically checked. If the checking process succeeds, an intermediate model called flat s-COMMA is generated, the aim of this model is to simplify the task of translators. Finally, the flat s-COMMA file is taken by the selected translator which generates the executable solver file.

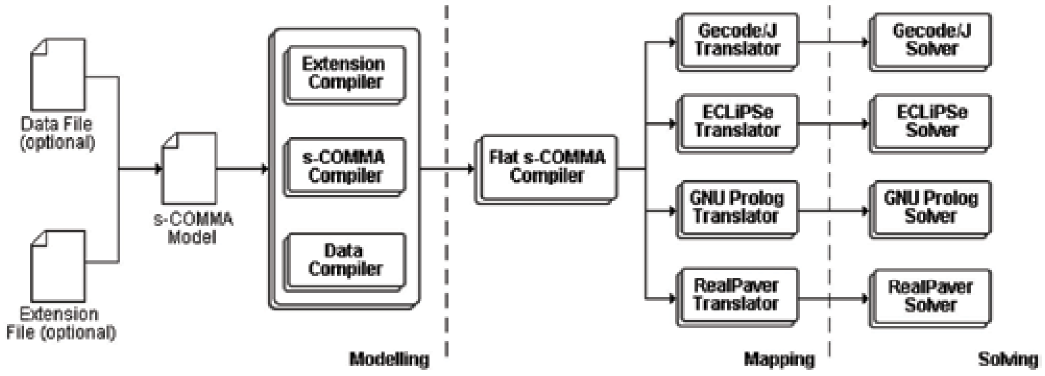


Fig. 7. s-COMMA architecture.

3.1 From s-COMMA to flat s-COMMA

A direct translation from s-COMMA to executable solver code is feasible (in fact, we have studied this in (Soto & Granvilliers, 2007)). However, many statements provided by s-COMMA are not supported by solvers. Thus, for performing this direct mapping, many model-transformations must be carried out at the level of translators. This makes translators bigger (in terms of code lines) and, as consequence, difficult to develop and maintain. A well-known technique to simplify code generation is to include an intermediate phase where the non-supported features are transformed to simpler (or supported) features. We state this transformation on an intermediate model called flat s-COMMA. The set of performed transformations from s-COMMA to flat s-COMMA are described below.

Flattening composition. The hierarchy generated by composition is flattened. This process is done by expanding each object declared in the main class adding its attributes and constraints in the flat s-COMMA file. The name of each attribute has a prefix corresponding to the concatenation of the names of objects of origin in order to avoid name redundancy.

Loop unrolling. Loops are not widely supported by solvers, hence we generate an unrolled version of loops.

Enumeration substitution. In general solvers do not support non-numeric types. So, enumerations are replaced by integer values, original values are stored to give the results.

Data substitution. Data variables are replaced by its value defined in the data file.

Conditional removal. Conditional statements are transformed to logical formulas. For instance, `if a then b else c` is replaced by $(a \Rightarrow b) \wedge (a \vee c)$.

Logic formulas transformation. Some logic operators are not supported by solvers. For example logical equivalence $a \Leftrightarrow b$ and reverse implication $a \Leftarrow b$. We transform logical equivalence expressing it in terms of logical implication. Reverse implication is simply inverted $b \Rightarrow a$.

Finally, the generated flat s-COMMA code is taken by the selected translator which generates the executable solver file.

4. Related work

s-COMMA is closely related to recent standard modelling language proposals as well as object-oriented languages for modelling constraint problems.

4.1 The definition of a standard modelling language

The definition of a standard modelling language for CP is a recent trend. First encouragements on this issue were done by J-F. Puget in (Puget, 2004). He suggested to develop a ``model and run'' paradigm such as in Math Programming. The paradigm involved a standard file format for expressing models and a CP library to solve them. Then, at *The Next 10 Years of CP* (Benhamou et al., 2007), this challenge was confirmed as an important research direction. Recently, at CP 2007 Conference, MiniZinc (Nethercote et al., 2007) was proposed as a standard modelling language. MiniZinc can be seen as a subset of elements provided by Zinc (Rafeh et al., 2007). The syntax is closely related to OPL (Van Hentenryck, 1999) and its solver-independent platform allows translating MiniZinc models into Gecode and ECLiPSe solver models. Essence (Fritsch et al., 2007) is another good basis to define such a standard. This core is focused on users with a background in discrete mathematics; this style makes Essence a specification language rather than a modelling language. The Essence execution platform allows mapping specifications into the ECLiPSe solver.

We believe s-COMMA may be a good starting point too, the constraint language of s-COMMA is closely related to OPL and Zinc. The solver-independent platform is an adequate support to map models to four different solvers. The object-oriented framework and the extensibility are also important features not present in the aforementioned proposals.

4.2 Objects + Constraints

The first attempt in combining an object oriented language with a constraint language was on the development of ThingLab (Borning, 1981) which was built for interactive graphical simulation. A next version of this approach was developed in the Kaleidoscope language (Freeman-Benson, 1992). Then, similar ideas were developed in Gianna (Paltrinieri, 1994) for modelling constraint-based problems with objects in a visual environment. COB (Bharat & Tambay, 2002) is a more recent approach for the engineering field, the language is a combination of objects first order formulas and CLP (Constraint Logic Programming) predicates. Modelica (Fritzson & Engelson, 1998) is another object-oriented system for modelling engineering problems, but it is mostly oriented towards simulation. In general, the constraint language of these approaches was designed to specific application domains. They also lack of extensibility and solver-independence.

5. Conclusion

In this chapter we have presented s-COMMA, a new language for modelling CSPs. The basis of s-COMMA is a combination of an object-oriented language with an extensible constraint language which allows users to state modular models using the benefits of the object-oriented style. s-COMMA is also supported by a three layered architecture where models can be mapped to four well-known solvers. We believe that these capabilities make s-COMMA a good basis to define a standard modelling language.

To reach this final purpose, several aspects could be developed, for instance: more work on benchmarks, solver cooperation, new global constraints and translation to new solvers. The development of a tool for modelling CSPs through a UML-like language will be useful too.

6. References

Benhamou, F.; Jussien, N. & O'Sullivan, B. (2007). Trends in Constraint Programming. ISTE, ISBN: 9781905209972, England.

Bharat, J. & Tambay, P. (2002). Modelling Engineering Structures with Constrained Objects. Proceedings *of Principles and Practice of Declarative Languages (PADL)*, pp. 28-46, LNCS Springer-Verlag, Portland, USA.

Borning, A. (1981). The Programming Language Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory. ACM Transactions on Programming Languages and Systems., 3(4): pp. 353-387, ISSN:0164-0925.

Schulte, C. & Stuckey, P. (2004). Speeding Up Constraint Propagation. *Proceedings of Principles and Practice of Constraint Programming (CP)*, pp. 619-633, LNCS Springer-Verlag, Toronto, Canada.

Diaz, D. & Codognet, P. (2000). The gnu prolog system and its implementation, *Proceedings of the ACM Symposium on Applied Computing (SAC)*, pp. 728–732, ACM Press, Villa Olmo, Italy.

Frisch, A.; Grum, M.; Jefferson, C.; Martínez, B. & Miguel, I. (2007). The design of Essence: A constraint language for specifying combinatorial problems, *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 80–87, Hyderabad, India.

Freeman-Benson, B. (1990). Kaleidoscope: Mixing Objects, Constraints and Imperative. *Proceedings of European Conference on Object-Oriented Programming (ECOOP)*, pp. 77-88, SIGPLAN Notices 25(10), Ottawa, Canada.

Fritzson, P. & Engelson, V. (1998). Modelica - A Unified Object-Oriented Language for System Modelling and Simulation. *Proceedings of European Conference on Object-Oriented Programming (ECOOP)*, pp. 67-90, LNCS Springer-Verlag, Brussels, Belgium.

Granvilliers, L & Benhamou, F. (2006). Algorithm 852: Realpaver: an interval solver using constraint satisfaction techniques. ACM Transactions on Mathematical Software., 32(1):pp.138-156, ISSN:0098-3500.

Nethercote, N.; Stuckey, P.; Becket, R.; Brand, S.; Duck, G. & Tack, G. (2007). Minizinc: Towards a standard cp modelling language. *Proceedings of Principles and Practice of Constraint Programming (CP)*, pp. 529–543, LNCS Springer-Verlag, Providence, USA.

Paltrinieri, M. (1995). A Visual Constraint-Programming Environment. *Proceedings of Principles and Practice of Constraint Programming (CP)*, pp. 499–514, LNCS Springer-Verlag, Cassis, France.

Puget, J. (1994). A C++ implementation of CLP. *Proceedings of the Second Singapore International Conference on Intelligent Systems*, Singapore.

Puget, J. (2004). Constraint programming next challenge: Simplicity of use. *Proceedings of Principles and Practice of Constraint Programming (CP)*, pp. 5–8, LNCS Springer-Verlag, Toronto, Canada.

Rafeh, R.; García de la Banda, M.; Marriott, K. & Wallace, M. (2007). From zinc to design model. *Proceedings of Principles and Practice of Declarative Languages (PADL)*, pp. 215–229, LNCS Springer-Verlag, Nice, France.

Soto, R. & Granvilliers, L. (2007). The Design of COMMA: An extensible framework for mapping constrained objects to native solver models. *Proceedings of the International Conference on Tools with Artificial Intelligence* (ICTAI), pp. 243–250, IEEE Computer Society, Patras, Greece.

Van Hentenryck, P. (1999). Constraint Programming in OPL. *Proceedings of Principles and Practice of Declarative Programming (PPDP)*, pp. 98-116, ACM Press, Paris, France.

Wallace, M.; Novello, S. & Schimpf, J. (1997). Eclipse: A platform for constraint logic programming, Technical report, IC-Parc, Imperial College, London, England.

Software Component Clustering and Retrieval: An Entropy-based Fuzzy *k*-Modes Methodology

Constantinos Stylianou and Andreas S. Andreou
Department of Computer Science, University of Cyprus
Cyprus

1. Introduction

Component-based software engineering is a stepwise software development process that relies on integrating small, independent software pieces into one larger fully-functioning system. However, a delay in any of the steps may negatively affect a project's schedule and, even worse, jeopardise its completion. Furthermore, given that the process is ultimately evaluation-oriented it is very difficult to find ways to accelerate certain stages in the process since developers are unwilling to compromise quality over time. Consequently, the only substantial means of reducing development time is to make components more easily and readily accessible to reusers in order to shorten the time taken for locating them. For this reason, component repositories have been introduced to store and organise software components. However, with an ever-increasing number of components, covering a wider variety of functionalities, there has been a parallel increase in the need for techniques to search and retrieve software components from repositories. Thus, the main motive of the research activity discussed in this chapter is to attempt to minimise the time necessary to discover any or all of the components to be reused in a component-based system by proposing an intelligent clustering methodology to aid the searching and retrieval of components stored in a repository.

To date there has been a number of attempts for clustering and classifying software components for reuse which have lead to the formation of various techniques. Several of these rely on structuring a formal specification of software components using classification methods based on natural language processing and documentation and retrieving components based on the resulting formal specification. Other attempts employ artificial intelligence methods for clustering software components, including genetic and evolutionary algorithms as well as self-organising feature maps. The methodology described in this chapter focuses on the utilisation of computational intelligence methods as the underlying clustering mechanism based on predefined lexical categories of software component attributes. Furthermore, the methodology incorporates a novel mechanism for efficiently searching and retrieving software components residing in a repository.

Specifically, the methodology employs an entropy-based fuzzy *k*-modes clustering algorithm, which has two main purposes. Firstly, to compute the number of clusters in a software component repository and to identify candidate initial cluster centres and secondly, to partition the software components into clusters with degrees of membership and to locate the final cluster centres. Accordingly, the methodology continues to compare a user's search preference with the final cluster centres in order to isolate the cluster closest to the preference. The methodology concludes with an innovative retrieval technique to return those components from within the isolated cluster that is closest to the search preference by taking into account the degree of participation of each component in that cluster.

The evaluation of the efficiency and effectiveness of the methodology was carried out with two tests. The first test aimed at examining whether, with a given set of software components, the clustering mechanism adequately grouped similar software components. The second test aimed at assessing whether the retrieval mechanism appropriately fetched the most suitable software components based on a user's search preference. Observations of the results of the experiments proved that the combination of an entropy-based algorithm with a fuzzy *k*-modes algorithm as an intelligent partitioning scheme efficiently clusters software components into groups, which is significantly important as it deals with the difficult issue of separating data into an unknown number of sets. Furthermore, the results also showed that the approach exhibits a strong filtering mechanism since it will always return the most suitable components from a repository for the user to evaluate and select which of these will be later integrated into the software system. Through the experiments it was also noticed that the parameters used in the methodology may significantly affect the clustering and retrieval of software components.

The remainder of this chapter is organised as follows: Section 2 presents an overview of the component-based software engineering development process and also looks into various previous attempts carried out to deal with the problem of software component clustering, classification and retrieval. Subsequently, Section 3 describes the main concepts of clustering and gives a detailed explanation of two clustering algorithms employed within a three-step fuzzy logic-based method proposed by (Tsekouras et al., 2005). Next, Section 4 describes the proposed methodology, which uses the algorithms explained in Section 3, but as applied to the problem of clustering software components in a repository. This section also gives a description of the mechanism developed for the retrieval of the most suitable software components based on a user's search preference. Continuing, Section 5 provides an analysis of the methodology's assessment procedure as well as an evaluation of the results obtained. Finally, in Section 6 the main strengths and weaknesses of the methodology are discussed in addition to possible alternatives or modifications that could be carried out in future research efforts.

2. Component-based software engineering overview

This section explores the component-based software development process and reviews various attempts and techniques that have previously been developed relating to the issue of component clustering and classification, in addition to searching and retrieval.

2.1 Software reuse, components & component-based software engineering

The idea of reuse is not recent. Ever since programming began, developers have been reusing previously implemented code segments, functions and procedures to build new software systems. However, it was not until 1968 at the NATO Software Engineering Conference at Garmisch, that the notion of "software reuse" was first formalised in an attempt to tackle the software crisis. There, Douglas McIlroy (Naur & Randell, 1969) argued that systems should be built from "reusable software components" and over the years there have been many definitions given for the term. One of which states that software components are "independent deployable software entities with a certain functionality which can be composed into larger systems by means of dynamically discoverable, immutable interfaces following standardised conventions" (Naedele, 2000). By this definition, it is possible for a software system to be comprised entirely of smaller, individual units, each of which performs a specific functionality in the system. Subsequently, the introduction of software components also gave rise to a new "integrate, not implement" style of software development named component-based software engineering. Systems built from reusable components following this approach are said to be component-based systems and over the years the software engineering industry has witnessed the expansion of the concept of "software reuse" to include not only code, but requirements, designs, architectures, test-cases and many more reusable assets of software engineering (Frakes, 2007).

2.2 Trends in component-based software engineering

Recent years have seen an increase in the standardisation and adoption of component reuse approaches to software development due to the shift of developers/programmers towards minimising the re-creation of code. For instance, object-oriented programming with its mechanisms for methods, inheritance and dynamic binding is one such approach that adopts reuse. Minimising the re-creation of code in turn has prompted software houses to prefer to employ component-based methodologies for developing systems over traditional ones, offering them a much simple and accessible solution. Furthermore, the component-based software development process has been improved with the introduction of component repositories for storing large volumes of software components.

With components becoming more readily available and having in mind that this will allow them to remain active in an ever-competitive industry, more companies are now willing to undergo the cost of buying software components. However, as explained in the next subsection, the process of development based on component reuse is a reasonably rigorous approach which emphasises the evaluation of components, their correct integration into the software system, and the components' maintenance.

2.3 The component-based software development process

As with any other disciplined software development process, there are certain steps that must be followed for successful software component reuse. Specifically, the component-based software development process primarily consists of four steps (Brown & Wallnau, 1996):

1. Component qualification (sometimes referred to as suitability testing)
2. Component adaptation
3. Assembling components into systems
4. System evolution

Component qualification is the first step and is considered as "a process of determining fitness for use of previously-developed components that are being applied in a new system context" (Haines et al., 2007). This step is the most determinative of the four, because the success of component reuse is largely based on whether the correct component is selected for reuse. It can be logically separated into two phases – discovery and evaluation –

whereby the former involves searching and retrieving the most suitable software components while the latter adopts criteria to examine the "non-technical" aspects such as quality, usability, developer's credibility and many more.

Once a component has been selected for reuse, the second step requires component adaptation to be carried out. Components may need to be configured before being integrated into the system given that they are generally built to be used in different contexts (Haines et al., 2007), which may lead to a greater chance of conflicts between components.

The third step of the process entails the components assembly into the system. This can be viewed as the integration of the components by binding the individual components with a well-defined infrastructure. The infrastructure forming the system can be based on several architectural styles, including, databases, blackboards, and the more popular, object request brokers (ORBs) (Haines et al., 2007).

The final step of the process is system evolution, similar to the concept of software maintenance in traditional development life-cycles. This step deals with the evolving and upgrading of a system by replacing erroneous and outdated components with newer and more advanced components. This unavoidably will require the latest components to be tested and validated before replacing the previous one. If a component needs replacing, but there is no updated version for it, then the process will reiterate to the first step to discover and evaluate components again (Haines et al., 2007). As a result, this step may end up being the longest with respect to the process as a whole.

The process explained above can only be considered productive if the time taken to successfully reuse a software component is less than the time it would take for a developer to implement the particular functionality from scratch – and this extends to the development of the system as a whole. This limitation coincides with one of the most challenging issues facing software houses, that is: being able to deliver projects within the predetermined deadline, and without exceeding the budget or compromising the system's quality.

This is the point to which attention is drawn as the methodology proposed aims to provide a solution for minimising the development time of component-based systems without having adverse affects on productivity, effectiveness, and, above all, quality. In order to minimise the development time, it is imperative to identify which parts of the process are the most time-consuming and in turn try to reduce the duration required to complete them. From the steps explained above, it can be observed that component discovery is one of the lengthiest stages in the process. This is because developers struggle to locate the most appropriate components to evaluate. If it is possible to find a way to reduce the time spent by developers searching and retrieving components, then component-based software development process will become more productive, efficient and reliable. Furthermore, more time can then be allocated to the other steps of the development process, for instance, to evaluate the retrieved components further on in the development process, whose duration cannot or, rather should not, be reduced. Also, more time will be available for the proper adaptation and assembling of the software components into the system's framework.

2.4 Previous attempts of component clustering, search & retrieval

Over the years, various attempts have been made to aid developers in their search and retrieval of software components for reuse.

Firstly, (Prieto-Díaz & Freeman, 1987; Prieto-Díaz, 1991) attempted to tackle the issue of software component classification with an informal method whereby experts extract facets from keywords contained in software components, and subsequently using these facets to describe technical and non-functional features of components. The method then continues to retrieve components by employing a weighted conceptual graph to match users' queries with software components taking into account the facets extracted by the experts. Another informal method includes natural language processing in queries for the retrieval of software components. Such techniques include semantic networks and free-text analysis for automatic keyword extraction (e.g., Girardi & Ibrahim, 1995; Sugumaran & Storey, 2003; Yao & Etzkorn, 2004). Formal methods, alternatively, aim to cluster and classify components by providing a specification for software components (Chu et al., 2000; Nakkrasae & Sophatsathit, 2002). Additionally, (Nakkrasae et al., 2004) expand the use of a formal specification for classification by combining a fuzzy subtractive clustering technique. A more recent attempt by (Chang et al., 2005) implements a scheme for the automatic clustering of use-cases, actors, classes and other elements of object-oriented modelling into candidate components.

There are also some noticeably distinct approaches that employ computational and artificial intelligence methods. For example, (Wang et al., 2004) use self-organising feature maps in order to cluster a software component catalogue, while (Andreou et al., 2006) make use of genetic and evolutionary algorithms to cluster software components. The proposed methodology deals with the issue of component clustering and retrieval from a similar perspective as the last two abovementioned studies. By employing computational intelligence and fuzzy logic techniques the methodology focuses on the construction, firstly of the principal clustering mechanism and secondly, of the mechanism required for searching and retrieving the most adequate components from repositories.

3. Clustering algorithms

Clustering is a form of unsupervised learning that aims to analyse and organise data into groups based on their similarity (Jain et al., 1999). For this reason, clustering is widely used in various problem-solving and decision-making applications, such as document retrieval, marketing research, genotype assignment, insurance fraud identification, image segmentation, and city-planning, to name a few. Over the years, there have been many methods and techniques developed to perform cluster analysis. The main types of clustering methods that exist include (Han & Kamber, 2000):

- Partitional clustering: This is one of the most common clustering methods, which aims to partition data into a finite number of clusters based on various distance measures and criteria. Examples include *k*-means, PAM and CLARANS.
- Hierarchical clustering: This method transforms a dataset into a hierarchical tree structure. Hierarchical clustering models can adopt a bottom-up (agglomerative) approach, such as in AGNES, which iteratively merges clusters into larger clusters. On the other hand, they can adopt a top-down (divisive) approach, for instance as in DIANA, by repeatedly splitting clusters into smaller clusters. In either approaches merging or splitting ceases when a termination condition is satisfied (Kauffman & Rousseeuw, 1990).
- Density-based clustering: This type of clustering takes into account the fact that data objects may form clusters with arbitrary shapes and requires a cluster to continue to

grow on condition that the number of objects assigned to it (i.e., its density) exceeds a minimum threshold value. Examples of density-based methods can be found in the DBSCAN (Ester et al., 1996) and OPTICS (Ankerst et al., 1999) models.

- Grid-based clustering: This approach creates a grid structure from a finite number of cells on the data object space and only needs to operate clustering on this structure rather than on all the data objects per se. For this reason it requires very little processing time. Some of the most popular grid-based clustering methods include CLIQUE (Agrawal et al., 1998), STING (Wang et al., 1997) and WaveCluster (Sheikholeslami et al., 2000).
- Model-based clustering: This form of clustering assumes that each cluster has its own model and attempts to determine the best fit of the data to that given model.

From the above list, it is apparent that clustering can take many forms and furthermore, it offers a wide variety of possible solutions to data mining issues. For example, it has the ability to discover trends in large datasets and can be used to identify potential outliers that are inconsistent with the remaining data.

In order to improve the process of searching and retrieving components it was decided that first and foremost it was imperative to partition the software components in a repository to reduce, or rather, eliminate less suitable components. The component clustering process, therefore, is the focal point of the suggested methodology, which adopts a strategy to cluster components of a repository based on a similar approach used by (Tsekouras et al., 2005). Here, however, the authors applied their technique to help in the analysis of cultural data, in particular, ethnocultural identity (Cushner & Brislin, 1997; Tsekouras et al., 2005) which is a major field in study of cross-cultural adaptation. The subsequent subsections take an in-depth look at the clustering algorithms adopted and present the advantages for their use in the suggested methodology as well as their underlying limitations.

3.1 Entropy-based clustering

Entropy-based clustering (Yao et al., 2000) is a technique that was first introduced in 1998, which seeks to arrange similar data objects into clusters based on their total entropy values. The goal of the technique is to traverse the dataset in just one pass to determine the number of clusters present and also to identify the location of the cluster centres. The basic idea is that if a data object has many surrounding objects, then its total entropy value will be relatively lower and can therefore be considered as a strong candidate for a cluster representative. The entropy value calculated for each data object is based on a predefined similarity measure. The data object achieving the lowest total entropy value is selected as the first cluster centre. At this point, the algorithm uses a parameter, β, representing a threshold of similarity or association (Yao et al., 2000). Any data objects with high similarity to the recently selected cluster centre (i.e., data objects with a similarity value higher than the threshold β) are removed from the dataset. The reasoning behind this is that since these data objects are similar they should stop being considered as potential clusters, and instead be assigned to the cluster they are highly similar to. Once these data objects are removed from the dataset, the number of clusters is increased and the data object with the next least total entropy value is selected and the procedure repeats until there are no objects left in the dataset.

3.1.1 Entropy-based clustering notations

Let $X = \{X_1, X_2, \ldots, X_n\}$ be a set of n data objects described by m attributes. The entropy value, H_{ij}, of two data objects, X_i and X_j, is defined by:

$$H_{ij} = -E_{ij}\log_2(E_{ij}) - (1-E_{ij})\log_2(1-E_{ij}) \tag{1}$$

where $i \neq j$. E_{ij} is a similarity measure between X_i and X_j and is calculated using:

$$E_{ij} = e^{-\alpha D_{ij}} \tag{2}$$

where D_{ij} is the distance between data objects X_i and X_j, and a is calculated automatically by:

$$\alpha = -\ln(0.5)/\bar{D} \tag{3}$$

where $\bar{D}$ is the mean distance among the pairs of data objects in a hyperspace. From Equation (1), the total entropy value of data object X_i with respect to all other data objects is computed as:

$$H_i = -\sum_{\substack{j=1 \\ i\neq k}}^{n}\left[E_{ij}\log_2(E_{ij}) - (1-E_{ij})\log_2(1-E_{ij})\right] \tag{4}$$

Finally, let $Z_l = \{Z_{l,1}, Z_{l,2}, \ldots, Z_{l,m}\}$ represent a cluster centre, which is assigned the data object that achieves the least entropy value after each iteration.

3.1.2 Entropy-based clustering algorithm

The entropy-based clustering algorithm is presented in Fig. 1.

```
Algorithm: Entropy-based Clustering

1. Select threshold of similarity, $\beta$, and set the initial number of clusters $c = 0$.
2. Determine the total entropy values $H$ for each data object in $X$ based on Equation (4).
3. Set $c = c + 1$.
4. Select the data object $X_{min}$ with the least entropy $H_{min}$ and set $Z_c = X_{min}$ as the $c^{th}$
   cluster centre.
5. Remove $X_{min}$ and all data objects having similarity with $X_{min}$ greater than $\beta$ from $X$.
6. If $X$ is empty then stop; otherwise go to step 3.
```

Fig. 1. Entropy-based clustering algorithm

The key reasons for using an entropy-based clustering algorithm are two-fold. It is effectively capable of computing the number of clusters in a given dataset, in addition to discovering which objects are candidates for cluster centres. Secondly, it is highly efficient due to the fact that the entropy values of data objects are required to be calculated only once and after a cluster centre is determined, data objects are iteratively removed from the dataset.

3.2 Hard and fuzzy *k*-modes algorithms

In 1998, (Huang, 1998) introduced the *k*-modes algorithm as an alternative method of clustering objects in a dataset. This partitional clustering method is specifically designed to handle categorical data, a limitation posed by the hard and fuzzy *k*-means algorithms. There are 3 basic modifications to the original algorithm. Firstly, the distance measure between two objects is altered to a simple matching of the attributes of the dataset as opposed to the squared Euclidean distance. Secondly, the cluster centres are defined by the modal value of each attribute instead of the mean value. Finally, computation of the modes utilises a frequency-based method applied on every category of the dataset's attributes. Despite these alterations, the clustering method still keeps the efficiency of the *k*-means algorithm, especially for large datasets. As previous, the objective of the algorithm is to minimise its cost function in order to separate the dataset into clusters.

3.2.1 *k*-modes notations

Clustering by *k*-modes adopts extremely similar notations from those of *k*-means clustering. However, since the attributes in the latter consist of numeric data – and consequently their domains – modifications are required to accommodate the use of categorical values. Specifically, let $X = \{X_1, X_2, \ldots, X_n\}$ be a set of *n* data objects. Each of these objects can be described by a set of *m* attributes $A_1, A_2, \ldots, A_m$, and each attribute, A_j, can take any value from the attribute domain $\mathrm{DOM}(A_j) = \{a_j^{(1)}, a_j^{(2)}, \ldots, a_j^{(nj)}\}$ where n_j is the number of category values possible for attribute A_j, for $1 \leq j \leq m$. Data objects can therefore be logically represented by a conjunction of attribute-value pairs $[A_{i,1} = x_{i,1}] \wedge [A_{i,2} = x_{i,2}] \wedge \ldots \wedge [A_{i,m} = x_{i,m}]$, where $x_{ij} \in \mathrm{DOM}(A_j)$, for $1 \leq j \leq m$ (Kim et al., 2004) and hence this is extended so that each X_i can be represented by a vector $[x_{i,1}, x_{i,2}, \ldots, x_{i,m}]$ for $1 \leq i \leq n$. Finally, let a cluster centre be symbolised by $Z_l = \{z_{l,1}, z_{l,2}, \ldots, z_{l,m}\}$ for $1 \leq l \leq k$. To minimise a cost function $F(W, Z)$ (Bezdek, 1980) therefore requires that:

$$F(W, Z) = \sum_{l=1}^{k}\sum_{i=1}^{n} w_{li}^{\alpha} d(Z_l, X_i) \tag{5}$$

subject to:

$$0 \leq w_{li} \leq 1, \quad 1 \leq l \leq k, 1 \leq i \leq n \tag{6}$$

$$\sum_{l=1}^{k} w_{li} = 1, \quad 1 \leq i \leq n \tag{7}$$

$$0 \leq \sum_{i=1}^{n} w_{li} \leq n, \quad 1 \leq l \leq k \tag{8}$$

where k ($\leq n$) is a predefined number of clusters, $W = [w_{li}]$ is a $k \times n$ partition matrix, $Z = \{Z_1, Z_2, \ldots, Z_k\}$ is the set of cluster centres, and $d(\cdot, \cdot)$ is some measure of distance between the two objects. The fuzziness exponent, $a \in [1, \infty)$ found in Equation (5) identifies whether hard ($a = 1$) or fuzzy ($a > 1$) *k*-modes clustering occurs.

3.2.2 The *k*-modes dissimilarity function

In contrast with *k*-means clustering, which calculates the dissimilarity of objects with the Euclidean norm, the dissimilarity function comprises of matching the category values of attributes comprising the objects. If two objects share the same category in an attribute, then

the distance for that attribute is assigned a value of zero, symbolising that there is no difference. Whereas if the objects do not have the same category value for an attribute, then the difference is allocated a value of one, indicating that a difference exists. By employing such generalised Hamming distance (Kohonen, 1987; Hunag, 1998), it is possible to see how close two objects are by summing the number of mismatches of category values. Hence, the smaller the number of mismatches, the nearer the objects are. The formal representation of the dissimilarity function is as follows:
Let $X_1 = [x_{11}, x_{12}, \ldots, x_{1m},]$ and $X_2 = [x_{21}, x_{22}, \ldots, x_{2m},]$ be two data objects of X defined by m attributes. The dissimilarity between the two objects, $d(X_1, X_2)$, is denoted by:

$$d\left(X_1, X_2\right) = \sum_{j=1}^{m} \delta\left(x_{1j}, x_{2j}\right) \tag{9}$$

where:

$$\delta\left(x_{1j}, x_{2j}\right) = \begin{cases} 0, & x_{1j} = x_{2j} \\ 1, & x_{1j} \neq x_{2j} \end{cases} \tag{10}$$

The dissimilarity function in Equation (9) is then used to (re)assign a data object to a cluster. Accordingly, in the case of the hard *k*-modes algorithm, if object X_i yields the shortest distance with centre Z_l in a given iteration, this is represented by setting the value at the nearest cluster to 1 and the values at the rest of the clusters to 0 in the partition matrix *W*. Formally, for $a = 1$:

$$\hat{w}_{li} = \begin{cases} 1, & \text{if } d\left(Z_l, X_i\right) \leq d\left(Z_h, X_i\right), \quad 1 \leq h \leq k \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

On the other hand, in the case of the fuzzy *k*-modes algorithm, for $a > 1$, the partition matrix *W* is given by:

$$\hat{w}_{li} = \begin{cases} 1, & \text{if } X_i = Z_l \\ 0, & \text{if } X_i = Z_h,\ h \neq l \\ \dfrac{1}{\sum_{h=1}^{k} \left[\dfrac{d\left(Z_l, X_i\right)}{d\left(Z_h, X_i\right)} \right]^{1/(\alpha-1)}}, & \text{if } X_i \neq Z_l \text{ and } X_i \neq Z_h,\ 1 \leq h \leq k \end{cases} \tag{12}$$

for $1 \leq l \leq k$, $1 \leq i \leq n$. This means that if a data object shares the same values for all attributes with a particular cluster centre, then it will be assigned wholly to that cluster and not at all to the rest. Conversely, if a data object is not completely identical to any of the cluster centres, then the data object is assigned a membership degree to each cluster, which also solves the constrained optimisation problem of Equation (7) (Tsekouras et al., 2005).
Equation (6), however, does not emphasise any importance of a category in an attribute. For example in a hospital database, the category value "YES" of the attribute "SMOKER" may

be considered more important than the category value "NO". For this reason, (He et al., 2007) provides various methods to assign weights not only to the attributes but also to the categories in attributes to properly reflect the reality of dissimilarity between data objects. Alternatively, in order to take into account the frequencies of the categories in the dataset, (Huang, 1998) proposed a chi-squared distance, which gives greater importance to less frequent categories of an attribute. Likewise, (San et al., 2004) suggest integrating the relative frequency of a category in the distance function. In either case, the partition matrix, $W = [w_{li}]$, is computed by using Equation (11) for hard *k*-modes clustering and Equation (12) for fuzzy *k*-modes clustering.

3.2.3 The *k*-modes update function

Updating the centres in *k*-modes clustering uses a frequency-based method that basically assigns the most frequent category of each attribute as representative of the cluster. It should be noted, however, that the newly computed mode of a cluster does not necessarily mean it is an element of that cluster. Strictly speaking, every cluster, $Z_l = [z_{l,1}, z_{l,2}, \ldots, z_{l,m}]$ for $1 \le j \le m$, is updated so that:

$$\hat{z}_{l,j} = a_j^{(r)} \in \mathrm{DOM}\left(A_j\right) \tag{13}$$

where, in the case of hard *k*-modes clustering, $a_j^{(r)}$ satisfies:

$$\left|\left\{ w_{li} \middle| x_{i,j} = a_j^{(r)},\ w_{li} = 1 \right\}\right| \ge \left|\left\{ w_{li} \middle| x_{i,j} = a_j^{(t)},\ w_{li} = 1 \right\}\right|,\quad 1 \le t \le n_j \tag{14}$$

for $1 \le l \le k$. It can be observed from Equation (13) that the mode is neither unique. It is possible for two categories to tie as the most frequent in an attribute. In such cases, the algorithm will arbitrarily assign the first category that achieves the maximum frequency (Huang & Ng, 1999) for the attribute in its cluster.

In fuzzy *k*-modes clustering, the general idea of the update function remains the same – the mode is decided based on the category achieving highest frequency. The difference lies in the computation of the frequency since there are no crisp values, (i.e., zeros or ones) in the partition matrix but membership values. For this reason the update function is represented as Equation (14) though $a_j^{(r)}$ must now satisfy:

$$\sum_{i, x_{i,j} = a_j^{(r)}} w_{li}^{\alpha} \ge \sum_{i, x_{i,j} = a_j^{(t)}} w_{li}^{\alpha},\quad 1 \le t \le n_j \tag{15}$$

By Equation (14) the category assigned to attribute A_j of cluster Z_l is the category which "achieves the maximum of the summation of w_{li} to cluster Z_l over all categories" (Huang & Ng, 1999). Nevertheless, it is still possible for two or more categories to tie as most frequent in the attribute.

3.2.4 The *k*-modes algorithm

The *k*-modes algorithm performs exactly like the *k*-means algorithm in that it attempts to minimise the cost function, firstly by fixing *Z* to determine *W* and secondly, fixing *W* to determine *Z*. The algorithm required to perform this is outlined in Fig. 2.

```
Algorithm: k-Modes Clustering

1. Select $k$ initial clusters randomly $Z^{(1)} = \{Z_1^{(1)}, Z_2^{(1)}, \ldots, Z_k^{(1)}\}$.
2. Determine $W^{(1)}$ by selecting either Equation (11) or Equation (12) for the type of k-
   modes clustering, such that $F(W, Z^{(1)})$ is minimised.
3. Set $t = 1$.
4. Determine $Z^{(t+1)}$ based on Equation (13) satisfying either Equation (14) for hard k-
   modes clustering or Equation (15) for fuzzy k-modes clustering, such that $F(W^{(t)},
   Z^{(t+1)})$ is minimised.
5. If $F(W^{(t)}, Z^{(t+1)}) = F(W^{(t)}, Z^{(t)})$ then stop; otherwise go to step 6.
6. Determine $W^{(t+1)}$ using the same equation used in step 2, such that $F(W^{(t+1)}, Z^{(t+1)})$ is
   minimised.
7. If $F(W^{(t+1)}, Z^{(t+1)}) = F(W^{(t)}, Z^{(t+1)})$ then stop; otherwise set $t = t + 1$ and go to step 4.
```

Fig. 2. The *k*-modes clustering algorithm

Since the *k*-modes clustering algorithm is adapted to function on categorical data, it is a suitable candidate for the clustering algorithm required for the methodology proposed in the next section. Furthermore, because of the disorderly and chaotic configuration of components in a repository, the fuzzy *k*-modes clustering algorithm is preferred over hard *k*-modes clustering to allow for levels of ambiguity in a repository's configuration.

4. Proposed methodology

This section presents the proposed methodology for the clustering and retrieval of software components from a component repository, indicating any necessary parameters as well as computed outputs. The three-step approach is defined by: (i) the clustering procedure including any pre-processing activities required, (ii) the input of the user's search preference and the isolation of the subset of software components in which the actual search will be carried out, and (iii) the retrieval of the most suitable components from the repository.

4.1 Software component attributes

Software components in repositories are composed of many different behavioural, functional, or structural attributes (Nakkrasae & Sophatsathit, 2002). The main attributes chosen to comprise the software components in the methodology were selected based on an earlier attempt carried out by (Andreou et al., 2006), which features software components described by both technical and non-functional characteristics. Specifically, the software component attributes are: component domain and specific functionality, platform, implementation language, operating system independence, synchronisation, visibility of implementation, price range, processor, memory and disk utilisation, binding, data encryption, data open format compatibility and finally, password protection. Furthermore, each of the 15 selected attributes is assigned a predefined domain that consists of categorical values (or categories). It is important to note that any other schemes attempting to employ the suggested methodology are not bound by the use of only these 15 attributes. In fact, other attributes may be included or present ones removed depending on the quality of the information found in the component repository.

4.2 Roles and parameters of the methodology

There are two main roles in the methodology. Firstly, there is that of the "application administrator" whose is responsible for configuring the nonuser-related parameters required by the clustering mechanism. The second role is that of the "user", who is required to provide their search preference and the level of confidence (covered in Subsection 4.5) to control the level to which the results should satisfy their search. Furthermore, as the approach is loosely based on a client-server model, the application administrator is involved at the server-side and conversely, the users' interactions and searches take place at the client-side.

4.3 Clustering procedure

The purpose of this step, as mentioned previously, is to avoid having to locate candidate components from the whole of the repository, but instead to isolate a part of it and perform the search in that part. Therefore, the first step in the methodology entails the clustering of the software components into groups based on the 15 features listed in Subsection 4.1. However, a problem arises since the number of groups to be formed is initially unknown. Hence, the clustering technique to be employed must be capable of determining the number of clusters in the repository automatically, while still being able to respond to and cope with an extremely large volume of high-dimensional data. The partitioning technique used therefore, utilises a clustering approach suggested by (Tsekouras et al., 2005), which consists of pre-processing data using the entropy-based clustering algorithm before feeding the results into the connecting fuzzy *k*-modes clustering algorithm.

4.3.1 Entropy-based clustering

The pre-processing begins by executing the entropy-based clustering algorithm explained in Section 3.1 on the repository in order to compute the number of clusters present as well as the candidate cluster centres. The entropy value of each component is calculated by evaluating the dissimilarity of each component with regards to all others. Here, the categorical nature of component attribute-types leads to using the Hamming distance as the most suitable dissimilarity measure between two components. Subsequently, a new cluster is formed assigning the component achieving the lowest entropy value. Next, all components achieving similarity equal or above the value of the threshold, β (selected by the application administrator) are assigned to the newly created cluster and cease from being considered as potential centres. At the end of this stage, the entropy-based clustering algorithm will have computed the parameter k along with the k initial cluster centres for the fuzzy *k*-modes algorithm to apply.

4.3.2 Fuzzy *k*-modes clustering

With the completion of the pre-processing stage, the clustering procedure executes the fuzzy *k*-modes clustering algorithm using as inputs the value of k and the initial cluster centres. The final input required is the fuzziness exponent, a, and this is selected by the application administrator in order to set the level of fuzziness to be considered in the clustering of the software components. For each component, the measure of similarity between each cluster centre is computed by using a simple pattern match of the attributes of the components to the cluster centres to determine the distance. Each component will then be assigned to all clusters. However, the higher the similarity to a cluster centre, the higher the participation (level of membership) the component will have in the respective cluster. After each iteration (i.e., after all components have been assigned), using the appropriate update functions of Equations (13)-(15), the cluster centres are revised resulting in new cluster centres. Specifically, the new centres are found by computing the mode from the categories of component attributes that achieve the highest summation of membership degrees in the cluster. The process of assignment and update repeats until the algorithm converges. The end results will be a partition matrix holding the membership degrees (easily convertible to percentages) of components to clusters as well as the finalised cluster centres. Both these elements will be used in the subsequent steps. Table 1 presents an example of a component partition matrix indicating the degree of membership of each component in all clusters. The values in boldface represent the highest membership degree of a component, relative to the cluster in which it achieved the degree. As shown in the matrix it is possible for a component to achieve highest membership in more than one cluster, as for example components C4 and C6.

	Component Assignments									
	C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**
Cluster Z1	0.13	0.50	**0.94**	**0.38**	0.17	**0.44**	**0.79**	0.04	0.02	0.13
Cluster Z2	0.08	**0.69**	0.03	0.18	0.17	0.09	0.10	**0.86**	0.03	0.08
Cluster Z3	0.22	0.20	0.01	0.06	**0.49**	**0.44**	0.01	0.06	**0.90**	0.22
Cluster Z4	**0.57**	0.24	0.03	**0.38**	0.17	0.03	0.10	0.04	0.05	**0.57**

Table 1. Example of a component partition matrix

4.4 Isolating the search cluster

After components have been assigned membership degrees to all the clusters and the final cluster centres have been determined, a user is able to search for components in the repository. The search procedure is not necessarily limited to one user at a time since each search only needs read-access to the cluster centres and membership degrees. Specifically, a user provides their preference through a simple user-interface application by selecting from the available categories in each attribute. The search preference is then transformed into a vector, just like in the form of the components in the repository, and is matched against the final cluster centres produced as a result of the previous clustering procedure. Subsequently, the cluster's index whose centre is nearest to the user's search preference is isolated to act as the "search cluster", based on the belief that it will inherently contain components nearer to the user's search preference. As always, the measure of closeness between a cluster centre and the search preference is calculated using the generalised Hamming distance. If a user decides not to give a category for certain attributes, then those attributes will not be taken into consideration during the matching with cluster centres and thus will be assessed based on a reduced attribute set. Furthermore, in the case where more than one cluster centre is nearest to the search preference (i.e., with equal distances from the search preference) then they are isolated and merged into one search cluster. Apart from their search preference, a user can optionally select a value for a level of confidence, which represents a cut-off limit on the ranking of subsequently retrieved components.

4.5 Retrieval of software components

The final step of the methodology is to retrieve components from the repository and display the results to the user. The search results will include components contained only in the search cluster as this is considered to store software components nearest to the user's search preference. Furthermore, it is important that the retrieved results are ranked in order to allow the user to make a more informed reuse decision. The retrieval mechanism first calculates the membership degree of the search preference with respect to the search cluster. Therefore, components with similar degrees of participation in the search cluster will be near to the user's search preference. Thus, by isolating a range around the preference's degree of membership, the software components further away from the search preference can be eliminated whilst keeping those that are nearer. This range is constructed by defining an upper and lower bound based on the value of the preference membership degree. For experimentation purposes, this was set at ±10% meaning that, for instance, if a search preference was assigned a membership degree of 60% in the search cluster, the components retrieved would have membership degrees between 50%-70%. An example of this is presented in Fig. 3. In cases where the upper bound or lower bound is calculated to be higher than 100% or 0% respectively, the bounds are truncated accordingly.
The next stage of the retrieval procedure takes the subset of components and matches them against the user's search preference to find all the distances between the search preference and each component in the subset. This is done to form a ranking (in percent) from highest to lowest of the components based on the user's search preference matching attributes over the total number of attributes. However, before displaying the software components to the user, the level of confidence (given by the user in the second step of the methodology) is applied to eliminate those components that fall below a suitable level of similarity selected by the user. After the ranked set of components has been formed, the results of the search are ready to be displayed to the user.

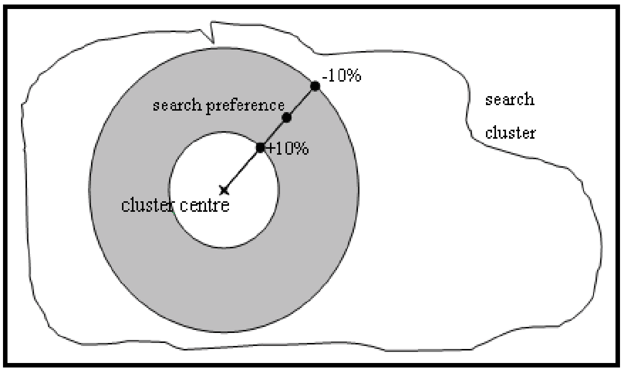


Fig. 3. Retrieval of the closest (suitable) components

The steps in the proposed methodology are outlined in Fig. 4. The methodology is shown as two separate units – one for the server (application administrator's) side and one for the client (user) side. Their only interaction is with the final cluster centres and the partition matrix. Of course, in reality, there would be more user instances connecting to the server.

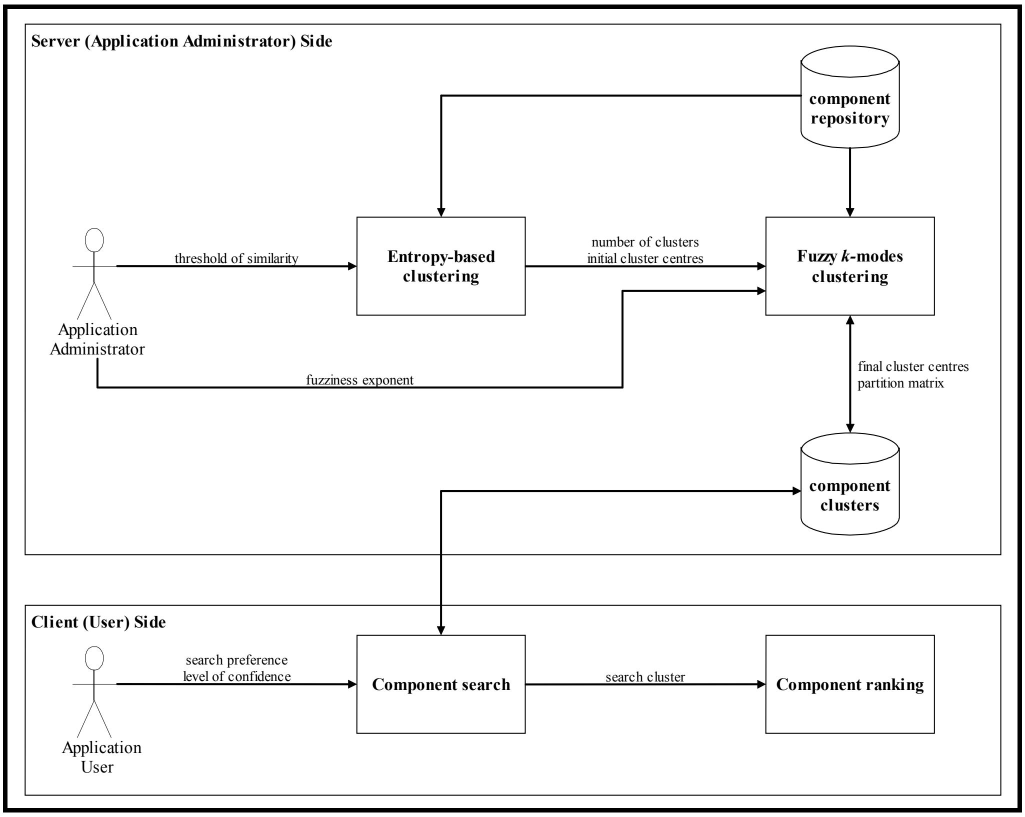


Fig. 4. Illustration of the proposed methodology

5. Experimental results

In order to test the efficiency and effectiveness of the methodology, a two-fold evaluation was carried, firstly to examine the clustering mechanism and secondly, to evaluate the retrieval mechanism.

5.1 Design of the experiments

The experiment involved the random generation of one-thousand software components and one random user search preference. For comparison reasons, the similarity of the search preference with all components was calculated and recorded, in order to distinguish which of these were closest to the search preference, as well as to be able to examine whether the clustering, isolation and retrieval processes do in fact return these closest components. The comparison was carried out according to the closest components in the dataset achieving similarity above: (1) 50% (near), (2) 75% (nearer) and (3) 90% (nearest). In addition, the methodology was tested by performing searches using three different variations of the preference: using all 15 attributes, 8 attributes and finally, 4 attributes.

For entropy-based clustering, an average threshold of similarity parameter (β = {0.50, 0.55}) was selected, bearing in mind the small number of attributes in the component dataset. Preliminary tests showed that with generally low total entropy values, a high threshold leads to cases were, once the selection of the component with the lowest entropy value forms a cluster centre, very few or no components would achieve a higher similarity to the threshold in order to be assigned it. In this instance, clusters will only be composed of a single component – the centre itself – resulting further in the formation of an undesirably high number of cluster centres. As regards to fuzzy *k*-modes clustering, in particular, its fuzziness exponent, it was decided to use values of a = {1.10, 1.20} to control the level of fuzziness. Less strict values (i.e., high exponents) during the clustering procedure may lead to the overlooking of some of the suitable components during retrieval, which again is an undesirable outcome.

A prototype tool to support the methodology was built for both the application administrator (responsible for the clustering procedure) and the users/developers (for searching and retrieving). The tool, as with the implementation of the algorithms, was built using Matlab® Release 14, version 7.0.

5.2 Experimental results

A summary of the results are presented in Tables 2-4, displaying the number of components retrieved from the repository compared to the known number of components similar to the search preference.

The first evaluation (presented in Table 2) used all 15 attributes of the search preference to retrieve the most suitable components. With a "near" (>50%) similarity, it was calculated (using a simple pattern matching method) that 17 components were similar to the search preference. The best results were obtained with values β = 0.50 and a = 1.10, whereby the methodology managed to retrieve 13 of the 17 components, equalling a 76% match.

Similarity to search preference > 50% Number of known near components 17			
β = 0.50		β = 0.55	
a = 1.10	a =1.20	a = 1.10	a =1.20
13 (76%)	9 (53%)	11 (65%)	8 (47%)

Similarity to search preference > 75% Number of known nearer components 7			
β = 0.50		β = 0.55	
a = 1.10	a =1.20	a = 1.10	a =1.20
7 (100%)	7 (100%)	7 (100%)	7 (100%)

Similarity to search preference > 90% Number of known nearest components 2			
β = 0.50		β = 0.55	
a = 1.10	a =1.20	a = 1.10	a =1.20
2 (100%)	2 (100%)	2 (100%)	2 (100%)

Table 2. Percentage of the closest components retrieved using 15 attributes

Furthermore, it was observed that the 4 components not retrieved were actually not as close to the preference and that the most suitable components were indeed returned. In the second case (similarity >75%) with 7 "nearer" components, all combinations managed to achieve a 100% match, retrieving all 7 components. Likewise, the 2 "nearest" components with respect to the search preference (similarity >90%) were both retrieved.

In the second evaluation only 8 of the 15 attributes of the search preference were used to retrieve components. In particular the attributes chosen were: domain functionality, specific functionality, operating system independence, implementation language, platform, visibility of implementation, price, and binding. The results are summarised in Table 3 below.

Similarity to search preference > 50% Number of known near components 45			
β = 0.50		β = 0.55	
a = 1.10	a =1.20	a = 1.10	a =1.20
12 (27%)	9 (20%)	10 (22%)	8 (18%)

Similarity to search preference > 75% Number of known nearer components 10			
β = 0.50		β = 0.55	
a = 1.10	a =1.20	a = 1.10	a =1.20
9 (90%)	8 (80%)	8 (80%)	8 (80%)

Similarity to search preference > 90% Number of known nearest components 2			
β = 0.50		β = 0.55	
a = 1.10	a =1.20	a = 1.10	a =1.20
2 (100%)	2 (100%)	2 (100%)	2 (100%)

Table 3. Percentage of the closest components retrieved using 8 attributes

By using only 8 attributes, it is observed that more components belong in the three similarity ranges (45 in >50%, 10 in >75% and 2 in >90%) than compared to using all 15 attributes. This is down to the fact that a fewer number of attributes decrease the likelihood of finding mismatches and hence greater distances. Moreover, from the results of Table 3, the percentage of "near" (> 50%) components with respect to the actual retrieved components has fallen substantially since retrieval is now based on a stricter matching. Also, although only 12 components were retrieved, these in fact were the ones that were closest to the preference, so in effect, the most suitable were not lost. At the two higher levels of similarity the best "nearer" and "nearest" components were almost always retrieved, with matches ranging from 80%-90% for similarity >75% and 100% for similarity >90% with the various combinations of parameters. Lastly, similarly to Table 2, the best results are obtained by using values of β = 0.50 and a = 1.10, which retrieve all 7 "nearest" components.

In the third and final evaluation, only the 4 attributes believed to be the most likely to be included in a search took part. These were the domain functionality, specific functionality, operating system independence, and implementation language. Table 4 shows the results obtained from the evaluation. It is immediately evident that the percentage of the "near" components retrieved is considerably low when compared to the other above experiments that used 15 or 8 attributes since the number of known components has increased. Specifically, even with the best β = 0.50 and a = 1.10 parameter values only 15 of the 87 (17%)

"near" components are retrieved and 9 of the 23 (39%) "nearer" components. Despite these low figures, in both cases, the components retrieved were still the ones closest to the preference. Also, all 7 of the "nearest" components were retrieved indicating at large that the methodology is capable of eliminating the less suitable components from the results at various levels of confidence.

Similarity to search preference > 50% Number of known near components 87			
β = 0.50		β = 0.55	
a = 1.10	a =1.20	a = 1.10	a =1.20
15 (17%)	9 (10%)	11 (13%)	8 (9%)

Similarity to search preference > 75% Number of known nearer components 23			
β = 0.50		β = 0.55	
a = 1.10	a =1.20	a = 1.10	a =1.20
9 (39%)	8 (35%)	8 (35%)	7 (30%)

Similarity to search preference > 90% Number of known nearest components 7			
β = 0.50		β = 0.55	
a = 1.10	a =1.20	a = 1.10	a =1.20
7 (100%)	6 (86%)	6 (86%)	6 (86%)

Table 4. Percentage of the closest components retrieved using 4 attributes

Based on the results of the various experiments it can be deduced that in order for the component retrieval process to be more accurate and reliable, the search must be carried out with as many, if not all the attributes defining the components. Nevertheless, this is attributable to the way in which the retrieval of the components works with respect to the quantity of the attributes – not quality, and certainly not due to the clustering procedure itself. Notably in the latter two experiments, although the number of components retrieved is low, they were still the most suitable according to their similarity to the search preference. As regards the retrieval of the "nearest" components, (similarity >90%), the methodology always manages to locate and display the best and most suitable components.

6. Conclusions

The number of software houses attempting to adopt a component-based development approach is rapidly increasing. However many organisations still find it difficult to complete the shift as it requires them to alter their entire software development process and philosophy. Furthermore, to promote component-based software engineering, organisations must be ready to promote reusability and this can only be attained if the proper framework exists from which a developer can access, search and retrieve a component. Hence, in the case of component-based software systems the ability to deliver software systems on time, within budget and with an acceptable level of quality is largely affected by the efficiency and effectiveness of the mechanisms employed for searching and retrieval of software components. Component repositories, even though having a large impact on promoting component-based software development as an organised and accessible means for searching and retrieving software components, can be difficult and time-consuming to handle due to the usually large volume of residing components.

This chapter aspired to provide a methodology to improve the current means of searching and retrieving software components from a repository. Specifically, the proposed methodology employed computational intelligence techniques that seek to cluster software components based on their similarity and then continuing to search for components combining the suggested clustering scheme with an innovative mechanism for isolating and retrieving the closest and most suitable software components based on the search preference provided by its user.

6.1 Benefits and limitations

The proposed methodology has several advantages. Firstly, the clustering procedure employs highly simple, efficient and accurate algorithms, namely, the entropy-based and fuzzy *k*-modes clustering algorithms. Also, the methodology's scheme is very flexible since it can be easily adopted for use with any component repository, provided that the correct encoding of component attributes is applied. Furthermore, it can accommodate for an even higher-dimensional component representation (currently with 15 attributes) or even with a completely modified feature set.

On the other hand, the methodology is not without its drawbacks. One disadvantage lies in the fact that in reality, the results of the clustering algorithms rely on the subjectivity of the person performing the clustering, that is, the application administrator. Furthermore, it is their decision whether to keep a computed clustering scheme or to re-compute the final cluster centres. Also, there may be some cases where entropy-based clustering creates many clusters due to a low similarity of components in a repository. This would lead to a larger number for *k* and thus many final clusters. In this situation it is probable that a user's preference will be similar to many cluster centres and could lead to the search cluster containing too many clusters with which the preference will be compared too.

6.2 Discussion on future work

The number of parameters required in the methodology is relatively small. The application administrator is required to select values for the threshold of similarity of the entropy-based clustering algorithm and the fuzziness exponent of the fuzzy *k*-modes clustering algorithm. On the other hand, the user is only required to provide their search preference and level of confidence applied to the retrieved results. Even with its small number of input parameters in the clustering process, the values can have a significant impact on the search and retrieval of software components. One possible approach to eliminate this high dependence is to attempt to automatically calculate the most suitable values of these parameters instead of them having to be defined by the application administrator. This may be achieved either through pre-processing of the component repository or with a validation scheme after the clustering takes place. In (Tsekouras et al., 2005) the authors have proposed a cluster validity index that determines the optimal number of clusters based on compactness and separation criteria.

In general, the proposed methodology is functional and promising. Although, at first glance of the results, it seems to "lose" components during retrieval, in essence these components are of lower similarity, and in fact, the methodology exhibits a strong filtering mechanism since it will always return the best (closest) suitable components from a repository for the user to evaluate and select which of these will be later integrated into the software system.

7. References

Agrawal, R.; Gehrke, J.; Gunopulos D. & Raghavan, P. (1998). Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 94-105, ISBN 0-89791-995-5, Seattle, WA, USA, June 1998, ACM Press, New York City

Andreou, A.S.; Vogiatzis, D.G. & Papadopoulos, G.A. (2006). Intelligent Classification and Retrieval of Software Components, *Proceedings of the Thirtieth Annual International Computer Software and Applications Conference, Vol. 2*, pp. 37-40, ISBN 0-7695-2655-1, Chicago, IL, USA, September 2006, IEEE Computer Society, Los Alamitos

Ankerst, M.; Breunig, M.M; Kriegel, H.-P. & Sander, J. (1999). OPTICS: Ordering Points to Identify the Clustering Structure, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 49-60, ISBN 1-58113-084-8, Philadelphia, PA, USA, June 1999, ACM Press, New York City

Bezdek, J.C. (1980). A Convergence Theorem for the Fuzzy ISODATA Clustering Algorithms, *IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE)*, Vol. 2, No. 1, January 1980, 1-8, ISSN 0162-8828

Brown, A.W. & Wallnau, K.C. (1996). Engineering of Component-based Systems, *Proceedings of the Second IEEE International Conference on Engineering of Complex Computer Systems*, pp. 414-422, ISBN 0-8186-7614-0, Montreal, Canada, October 1996, IEEE Computer Society, Los Alamitos

Chang, S.H.; Han, M.J. & Kim, S.D. (2005). A Tool to Automate Component Clustering and Identification, *Proceedings of the Eighth International Conference on Fundamental Approaches to Software Engineering*, pp. 141-144, ISBN 978-3-540-25420-1, Edinburgh, Scotland, April 2005, Springer-Verlag, Berlin

Chu, W.C.; Lu, C.-W.; Yang, H. & He, X. (2000). A Formal Approach for Component Retrieval and Integration Analysis, *Journal of Software Maintenance: Research & Practice*, Vol. 12, No. 6, November/December 2000, 325-342, ISSN 1532-060X

Cushner, K. & Brislin, R.W. (1997). *Improving Intercultural Interactions: Modules for Training Programs, Vol. 2*, ISBN 978-0761905370, Sage Publications, California

Ester, M.; Kriegel, H.-P.; Sander, J. & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 226-231, ISBN 1-57735-004-9, Portland, OR, USA, August 1996, AAAI Press, Menlo Park

Frakes, W.B. (2007) Software Reuse, ReNews – Software Reuse and Domain Engineering, http://frakes.cs.vt.edu/renews.html

Girardi, M.R. & Ibrahim, B. (1995). Using English to Retrieve Software, *Journal of Systems & Software (Elsevier)*, Vol. 30, No. 3, September 1995, 249-270, ISSN 0164-1212

Capt. Haines, G.; Carney, D. & Foreman, J. (2007). Component-based Software Development/COTS Integration, *Software Technology Roadmap*, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, PA, USA, http://www.sei.cmu.edu/str

Han, J. & Kamber, M. (2000) *Data Mining: Concepts and Techniques*, Morgan Kauffman, ISBN 1-55860-489-8, CA, USA

He, Z.; Xu, X. & Deng, S. (2007). Attribute Value Weighting in *k*-Modes Clustering, *Computer Science e-Prints: arXiv:cs/0701013v1 [cs.AI],* Cornell University Library, Cornell University, Ithaca, NY, USA, http://arxiv.org/abs/cs/0701013v1

Huang, Z. (1998) Extensions to the *k*-Means Algorithm for Clustering Large Datasets with Categorical Values, *Data Mining and Knowledge Discovery (Springer)*, Vol. 2, No. 3, September 1998, 283-304, ISSN 1384-5810

Huang, Z. & Ng, M.K. (1999). A Fuzzy *k*-Modes Algorithm for Clustering Categorical Data, *IEEE Transactions on Fuzzy Systems (IEEE)*, Vol. 7, No. 4, August 1999, 446-452, ISSN 1063-6706

Jain, A.K.; Murty, M.N. & Flynn, P.J. (1999). Data Clustering: A Review, *ACM Computing Surveys (ACM)*, Vol. 31, No. 3, September 1999, 264-323, ISSN 0360-0300

Kauffman, L. & Rousseeuw, P.J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley-InterScience, ISBN 978-0471878766, NY, USA

Kim, D.-W.; Lee, K.H. & Lee, D. (2004). Fuzzy Clustering of Categorical Data Using Fuzzy Centroids, *Pattern Recognition Letters (Elsevier)*, Vol. 25, No. 11, August 2004, 1263-1271, ISSN 0167-8655

Kohonen, T. (1987). *Content-Addressable Memories*, Springer-Verlag, ISBN 038717625X, NJ, USA

Naedele, M. (2000). Presentation Slides for Component Software: An Introduction, *Industrial Software Systems CHCRC.C2*, ABB Corporate Research Limited, Baden, Switzerland

Nakkrasae, S. & Sophatsathit, P. (2002). A Formal Approach for Specification and Classification of Software Components, *Proceedings of the Fourteenth International Conference on Software Engineering & Knowledge Engineering*, pp. 773-780, ISBN 1-58113-556-4, Ischia, Italy, July 2002, ACM Press, New York City

Nakkrasae, S.; Sophatsathit, P. & Edwards, Jr., W.R. (2004). Fuzzy Subtractive Clustering-based Index Approach for Software Components Classification, *International Journal of Computer & Information Science (ACIS)*, Vol. 5, No. 1, January 2004, 63-72, ISSN 1525-9293

Naur, P. & Randell, B. (1969). Software Engineering, *Report of a Conference Sponsored by the NATO Science Committee*, Garmisch, Germany, October 1968, Scientific Affairs Division, NATO, Brussels, Belgium

Prieto-Díaz, R. & Freeman, P. (1987). Classifying Software for Reuse, *IEEE Software (IEEE)*, Vol. 4, No. 1, January 1987, 6-16, ISSN 0740-7459

Prieto-Díaz, R. (1991). Implementing Faceted Classification for Software Reuse, *Communications of the ACM (ACM)*, Vol. 34, No. 5, May 1991, 89-97, ISSN 0001-0782

San, O.M.; Huyini, V.-N. & Nakamori, Y. (2004). An Alternative Extension of the *k*-Means Algorithm for Clustering Categorical Data, *International Journal of Applied Mathematics and Computer Science (AMCS)*, Vol. 14, No. 2, 2004, 241-247, ISSN 1641-876X

Sheikholeslami, G.; Chatterjee, S. & Zhang, A. (2000). WaveCluster: A Multi-resolution Clustering Approach for Very Large Spatial Databases, *International Journal on Very Large Data Bases (Springer)*, Vol. 8, No. 3-4, February 2000, 289-304, ISSN 1066-8888

Sugumaran, V. & Storey, V.C. (2003). A Semantic-based Approach to Component Retrieval, *The DATA BASE for Advances in Information Systems (ACM SIGMIS)*, Vol. 34, No. 3, August 2003, 8-24, ISSN 0095-0033

Tsekouras, G.E.; Papageorgiou, D.; Kotsiantis, S.; Kalloniatis, C. & Pintelas, P. (2005). Fuzzy Clustering of Categorical Attributes and its Use in Analyzing Cultural Data, *International Journal of Computing Intelligence (WASET)*, Vol. 1, No. 2, January 2005, 123-127, ISSN 1304-2386

Wang, W. ; Yang, J. & Muntz, R. (1997). STING: A Statistical Information Grid Approach to Spatial Data Mining, *Proceedings of the Twenty-third International Conference on Very Large Data Bases*, pp. 186-195, ISBN 978-1558604704, Athens, Greece, August 1997, Morgan Kaufmann, San Francisco, California

Wang, Z.; Liu, D. & Feng, X. (2004). Improved SOM Clustering for Software Component Catalogue, *Proceedings of the International Symposium on Neural Networks, Vol. 1*, pp. 846-851, ISBN 978-3-540-22843-1, Dalian, China, August 2004, Springer-Verlag, Berlin

Yao, H. & Etzkorn, L. (2004). Towards a Semantic-based Approach for Software Reusable Component Classification and Retrieval", *Proceedings of the ACM Forty-second Annual Southeast Regional Conference*, pp. 110-115, ISBN 1-58113-870-9, Huntsville, AL, USA, April 2004, ACM Press, New York City

Yao, J.; Dash, M.; Tan, S.T. & Liu, H. (2000). Entropy-based Fuzzy Clustering and Fuzzy Modeling, *Fuzzy Sets and Systems (Elsevier)*, Vol. 113, No. 3, August 2000, 381-388, ISSN 0165-0114

24

An Agent-Based System to Minimize Earthquake-Induced Damages

Yoshiya Takeuchi, Takashi Kokawa, Ryota Sakamoto, Hitoshi Ogawa and Victor V. Kryssanov
College of Science and Engineering, Ritsumeikan University
Japan

1. Introduction

Over 2000 earthquakes happen every year in Japan, so that this country is often called earthquake-ridden (Government of Japan, 2006). There exists a serious problem to prevent the occurrence of earthquake-induced disasters, such as fire, short-circuits, gas leakage, etc. With the recent advent of nation-wide telecommunication networks, real-time earthquake information can be received at every household, and it can thus be utilized to control consumer electronics and reduce the risk of these earthquake-provoked disasters (Kueppers, 2002).
Real-time information about the seismic activity in Japan is provided by the national earthquake early warning system operated by the Japan Meteorological Agency, JMA (Doi, 2002). By sensing primary waves, this system can notify people (e.g. via radio and TV channels, a mobile phone subscription service, etc.) several seconds before the earthquake devastating secondary waves hit a specific area.
The Japan Electronics and Information Technology Industries Association (JEITA) has recently introduced an automatic consumer electronics control system (JEITA, 2005). When an earthquake early warning is received, this system provides services, such as activating alarms, stopping gas, opening doors, and the like. This system cannot, however, comply with the specific situation at each particular household, as circumstantial information about who live/stay in, where they are, what they currently do, etc. is not utilized by the system.
In the presented study, we propose an agent-based system for the earthquake-induced disaster prevention. The system uses household-specific knowledge and can provide for generally a higher level of safety for the inhabitants than existing systems with similar goals do. The proposed system realizes a distributed architecture – a design solution making it quite reliable in (post-)earthquake conditions. There are specialized agents installed in different places (e.g. of a house or a public facility) and called “room agents,” which are autonomous, monitor various appliances and people in the rooms, and can control the equipment and electronics, and guide evacuation when an earthquake happens. For the control, countermeasure agents processing different types of rules are set up. During an earthquake, a countermeasure agent receives earthquake data and selects appropriate constraints, which are to be used by the room agents. As there can often be conflicts when

simultaneously applying control rules obtained from different countermeasure agents, the system implements a conflict-resolving mechanism to produce an optimized set of rules by solving a weighted constraint satisfaction problem with achievement parameters.
In the chapter's remainder, the earthquake early warning system is first outlined. The architecture of a system prototype developed by the authors is then presented. Next, it is explained how the developed system agents act. A case study of the control of home appliances is described. The ability of the system to guide evacuation in an earthquake situation is analyzed through several simulation experiments. Finally, related work is briefly discussed, and conclusions are drawn.

2. The early warning system

All earthquakes produce two types of shock waves: primary (P) and secondary (S). P-waves arrive first and usually do not cause any damages. S-waves follow P-waves, are much stronger, and often result in devastation and loss of lives. The earthquake early warning system operated by the JMA deals with current seismic information, such as magnitude of an earthquake and place of its occurrence, obtained by sensing and processing data of the P-waves. Since P-waves are propagated about twice as fast as S-waves (excepting for the case of epicentral earthquakes), the system can usually provide earthquake information to its clients seconds to tens of seconds before the damaging wave hits an area.
In the presented study, a program developed by the Japan Weather Association and the Earthquake Research Institute at the University of Tokyo is used to calculate the expected seismic intensity and time of the S-wave arrival at a specific location for a given earthquake, based on the earthquake early warning data (Kikuchi, 2004).

3. The agent-based system

3.1 System architecture

Fig. 1 shows the architecture of the system proposed in this study. An earthquake information agent (EIA) is a "JAVA wrapping" of the program processing earthquake early warning data. The EIA receives an earthquake early warning from the JMA and calculates the S-wave arrival time and the expected seismic intensity. The EIA then communicates to three countermeasure agents: an earthquake countermeasure agent (ECA), a personal care agent (PCA), and a precondition for consumer electronics control agent (PCCA). The ECA utilizes general rules for earthquake disaster prevention. The PCA applies personalized rules by utilizing inhabitant-related information. The PCCA makes use of appliance-specific rules to appropriately control consumer electronics and other equipment in the room. The countermeasure agents propose constraints to room agents. After a room agent communicates (or attempts to communicate) to the countermeasure agents to update its rules, it generates, through resolving achievement-weighted constraints, a set of instructions to control the appliances and, possibly, to guide the evacuation process.

3.2 Human status and the system interface

Fig. 2 shows the experimental environment – a living room – used in the study. The room space is divided into 9 locations and a corridor (location 10) with spotlights to help navigate

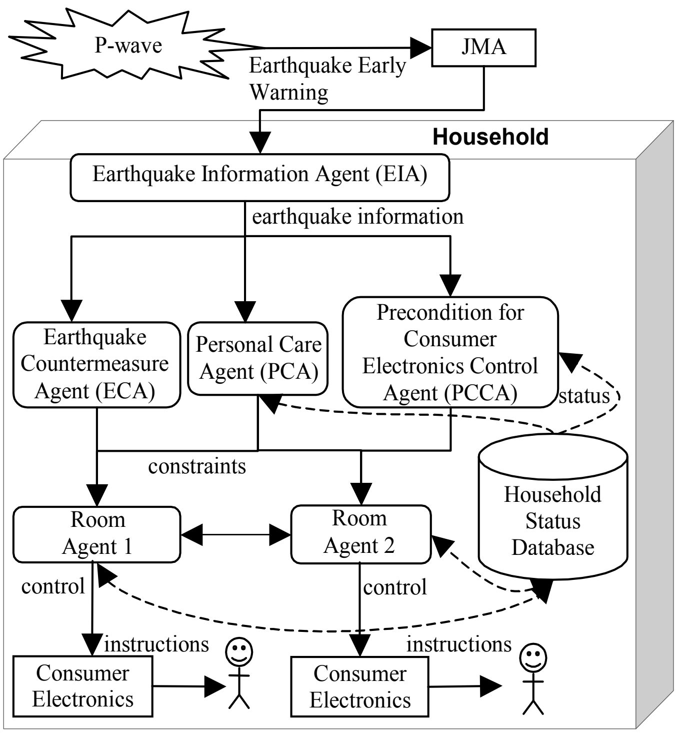


Fig. 1. System architecture

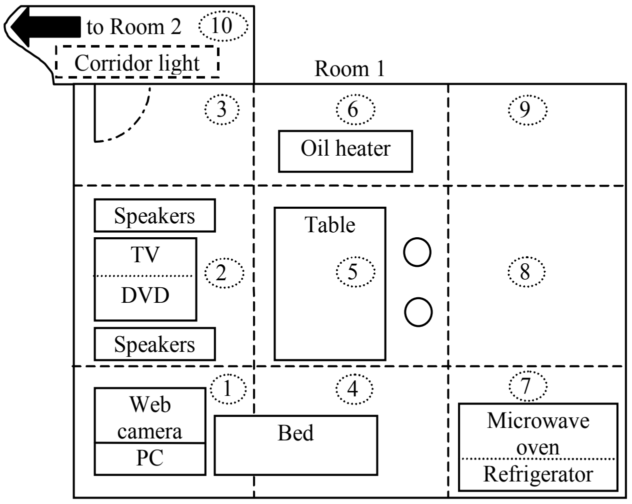


Fig. 2. Experimental environment (Room 1)

people. Current human and consumer electronics statuses are stored in the Household Status Database. Tables 1 and 2 list typical records of the database, which are continuously updated while the room agents monitor the environment. Table 1 gives an example of human statuses. The human status domain is a list of generally expected behavior, which is created, based on a personal profile (e.g. a healthy young man is expected to be able to help other people in the house, while someone with a badly injured leg could hardly move without assistance). Actions recommended by the system are selected from the corresponding pre-defined domains. (The difference in the domains shown in Table 1 is due to the difference in the individuals' current statuses.)

There are two types of human statuses processed by the system: static and dynamic. The static status is a general, "permanent" (or rarely changed) description of an individual present in the room: the individual's gender, age, physical abilities, etc. The dynamic status is regularly updated information about the individual's current behavior (e.g. sleeping/resting, being involved in noisy activities, etc.) and location. The dynamic status information is obtained by using RFID tags and an image-recognition system (with a camera connected). For example, when a person registered in the household database (i.e. someone with a static status record) enters the room, her or his location data of the dynamic status is updated, as the person approaches the corresponding sensors of the room agent. Any individual having no static status is automatically associated with a temporary "visitor/guest" profile generated by the system.

A consumer electronics status (Table 2) is predefined at the time when the corresponding appliance is installed. A domain for the control of the electronics and other equipment is also predefined (e.g. by downloading relevant rule-sets via a consumer electronics network), based on the appliance type and manufacturer. The current statuses of the appliances are updated when the electronics are controlled, whether automatically or manually.

When an earthquake early warning is received, room agents resolve achievement-weighted constraints, and a status rule-set for the consumer electronics and people in the room is determined. The system can change the electronics statuses by sending control signals (e.g. via infrared channels). Human behavior cannot, however, be controlled as such, and the system instead issues instructions, based on the most recently registered (static and dynamic) human status. At this point, the current status is (attempted to be) recognized with the sensors, and the records may be updated in the database.

Individual's ID#	Current (dynamic) status	Location (Room)	Domain for the human behavior variables
1	Watching (DVD), low activity	5 (1)	{Normal, Sleeping, Watching(X), Hiding under(X), Staying away from(dangerous object), Being accompanied, Making a contact, Reporting own location}
2	Sleeping, no activity	1 (2)	{Normal, Sleeping, Watching(X), Being accompanied, Making a contact, Reporting own location}

Table 1. Human status

The system keeps continuously updating the status database and producing control and evacuation instructions, based on the latest available information about the dynamically changing environment and the human behavior. The system thus realizes a "latent" interface for consumer electronics and other controlled equipment (e.g. oil or/and gas heaters, doors, etc.) by sensing and processing not only the early warning information received from the EIA, but also the response (of both the inhabitants and the electronics) to the instructions issued by the system, which is registered by the room agents. The interface is self-adapting (in the sense that its current state mainly depends on its previous state) and proactive (in the sense that it tries to minimize catastrophic consequences of earthquake-induced problems, which may arise in the future).

Appliance / equipment	Current status	Location	Domain of the control variables
TV	On(DVD)	2	{On(TV), On(DVD), On(CH *num*), Off}
DVD	Play(DVD)	2	{On, Play(DVD), Rec(CH *num*), Off}
Speaker	On(DVD)	2	{On(TV), On(DVD), On(PC), Announce(X), VolUp, VolDown, Off}
Light 1	Off	1	{On, Off}
Light 2	On	2	{On, Off}
Light 3	Off	3	{On, Off}
Light 4	Off	5	{On, Off}
Corridor light	Off	10	{On, Off}
Web-camera	Off	1	{Record, On, Off}
Heater	On	6	{On, Off}
Phone	Off	1	{Connect(Person *h*), Off}
Micro-wave oven	Off	7	{On, Off}
Refrigerator	On	7	{On, Off}

Table 2. Consumer electronics status (Room 1)

3.3 Constraints for the control

A room agent determines appropriate (optimized) control instructions by solving an achievement-weighted constraint satisfaction problem, AWCSP (Kokawa & Ogawa, 2004). An AWCSP solver implemented in the system is an enhanced reasoning engine for constraint satisfaction problems (CSP) that allows for obtaining a Pareto optimal solution even when the invoked constraints are too strict. The AWCSP is similar to the classic CSP (Walliser & Branschen, 2004) and is represented with a set of variables, a domain of values for each variable, and a set of constraints, but it also requires a set of constraint weights and a set of constraint achievement degrees defined. Various theoretical and applied aspects of AWCSP were actively studied in the past 10 years (Bistarelli et al., 1999; Luo et al., 2002; Schiex et al., 1995), as it became a relatively popular reasoning technique for agent-based systems (Yokoo, 2001).

In the developed system, consumer electronics statuses and human actions are represented as follows: a set of consumer electronics states, $CE = \{ce_1, ce_2, ce_3, \ldots, ce_n\}$, where n is the number of appliances installed; a set of human actions, $ACT = \{act_1, act_2, \ldots, act_m\}$, where m is the number of the inhabitants. Domains of the variables are represented as $D_{CE} = \{d_{ce_1}, d_{ce_2}, \ldots, d_{ce_n}\}$ and $D_{ACT} = \{d_{act_1}, d_{act_2}, \ldots, d_{act_m}\}$, respectively.

The ECA handles general countermeasure rules for earthquake disaster prevention, which are usually pre-defined. The PCA deals with human-related rules by utilizing information about the current status of each inhabitant. The PCCA processes specialized rules, based on a control policy defined for the household. The countermeasure agents produce constraints for the variables by utilizing the relevant rules. In a room agent, a set of constraints is represented as $C = \{c_1, c_2, \ldots, c_k\}$, k is the number of constraints. Below, these are examples of the knowledge of countermeasure agents:

```
ECA:
Rule: If (Seismic intensity >= 4) and
         (There is a person h in Room r)
     Send a Constraint to Room r agent:
         ECc1 {act_h = "Hide under furniture"}
PCA:
Rule: If (Seismic intensity >= 3) and ("There is a child at home")
         and (There is a person h(with activity) in Room r)
     Send a Constraint to Room r agent:
         PCc1 {act_h = "Accompany the child"}
PCCA:
Rule: If (Seismic intensity >= 5) and ("Agreement for recording")
     Send a Constraint:
         PCCc1 {state(ce_i, Camera), ce_i="Record", i = 1, …, n}
```

PCA also defines rules to support people with disabilities. For example, for those with hearing disabilities, important information is delivered in a visual form, e.g. on a TV screen or simply with blinking light. Analogously, whenever people with reduced vision are present in the room, all important information is delivered via voice and sound channels.

Constraints generated by the agents may often be in conflict. For example, if there is no furniture to hide under, a constraint "Hide under furniture" would never be satisfied, and no instructions would be issued by the system. The room agent should still recommend an accomplishable action for the inhabitants, such as, for example, "Stay away from dangerous objects." If one then tries to represent the expected behavior with weighted constraints, the rules become complicated and difficult to maintain the integrity (Yokoo, 2001). To cope with this problem, achievement degrees are defined for the constraints. When a constraint is fully satisfied, the room agent chooses a variable value having the highest achievement degree.

An achievement degree is a parameter showing when and to what extent a given constraint is satisfied. This parameter consists of a variable and a threshold. A set of the achievement variables $A = \{a_1, a_2, \ldots, a_k\}$ represents satisfaction degrees of the constraints. A set of achievement thresholds $F = \{f_1, f_2, \ldots, f_k\}$ gives thresholds that the values of a_i, $i = 1, \ldots, k$, must achieve to make the constraints satisfied.

An example of achievement degrees for a constraint, which refers to issuing recommendations that would help maintain a higher safety level for the inhabitants, is given in Table 3.

Achievement degree	Safety Level	Actions
a_i = (achievable safety level / recommended safety level by constraint c_i) × 100	5	Hide under furniture (table, etc.)
	4	Stay away from dangerous objects (window, etc.)
	3	Accompany someone
	2	Make a contact (via mobile, etc.)
	1	Report location
	1	No action required

Table 3. Achievement degrees and recommended actions

The assignment of values to the variables is done via an optimization procedure. Room agents produce sets of variable values corresponding to achievement degrees higher than the achievement threshold of a constraint. Room agents then calculate *M*, an "optimization degree" of the whole set as the following sum:

$$M = \sum_{i=1}^{k} \begin{cases} w_i, & f_i = 100\,, \\ w_i \times \dfrac{a_i - f_i}{100 - f_i}, & f_i < 100\,. \end{cases} \tag{1}$$

If a constraint c_i is completely satisfied, the corresponding summand is the constraint's weight w_i. Otherwise, the added value is a product of the constraint's weight w_i and the normalized distance between the achievement degree value a_i and the achievement threshold f_i; k is the total number of constraints. Of course, there may be used formulas other than the above to calculate *M*, depending on the optimization strategy chosen (e. g. see Schiex et al., 1995).

4. Case study

In this section, we describe a case study of the development of (a prototype of) the system shown in Fig. 1 for the environment specified in Tables 1-2 and (partly) in Fig. 2. Table 4 exemplifies situations considered in the case study, and Table 5 lists constraints and constraint parameters for the situations.

Situation 1 is, perhaps, the most usual (or expected) situation: Person 1 is relaxing in the living room, while a middle-level earthquake occurs; there is enough time for the action stipulated by c_2.

Situation 2: Person 1 is sleeping, and the system needs to awake her or him as specified with c_7.

Situation 3: the difference between Situations 3 and 2 is that there is not enough time for action in the former case. The system cannot advise to move to a safer place, as it would very likely cause panic rather than improve the overall safety.

Situations 4 and 5 show possible interactions between Room 1 and Room 2 agents, as assumed with c_4. There is a conflict between c_2 and c_4, since there is no furniture to hide under in Room 2 (this room is not shown in Fig. 2 but is assumed to be a Japanese-style *tatami* room with no furniture; see Fig. 4.). If Person 1 does not go to Room 2 but, instead, hides under the table, Person 1's own safety is maintained, but the safety of Person 2 with no activity is not. On the other hand, if Person 1 goes to Room 2 and accompanies Person 2, the own safety of Person 1 cannot be maintained sufficiently high, but the safety level of Person 2 is improved. In this experiment, the weights of c_2 and c_4 are the same, and the constraint achievement thresholds f_2 and f_4 are balanced to help the person with no activity, depending on the seismic intensity and the remaining time. Every household member is assumed to have a mobile phone.

Situation 6: there is a specified rule in the household policy. Constraint c_8 allows for making a video record for rescue operations or future analysis. Owing to privacy issues, the agreements of this type's constraints depend on the household privacy policy.

Situation number	Person 1 status (in Room 1)	Person 2 status (in Room 2)	Electronics Status	Early warning information	Explanation
1	Watching a movie	- (no data)	TV and DVD are turned ON	Seismic intensity is 4; Remaining time is 20sec	The system needs to get attention of Person 1 for announcing instructions
2	Sleeping, no activity	- (no data)	Refrigerator and micro-wave oven are turned ON (stand-by)	Seismic intensity is 4; Remaining time is 22sec	The system needs to wake Person 1 up (alarm and room light ON)
3	Sleeping, no activity	- (no data)	Refrigerator and micro-wave oven are turned ON (stand-by)	Seismic intensity is 5; Remaining time is 4sec	There is not enough time to sensibly act (e.g. escape / hide anywhere)
4	Reading a book	Sleeping, no activity	Room lights are turned ON	Seismic intensity is 4; Remaining time is 21sec	Person 1 would (attempt to) help Person 2 in Room 2
5	Watching a movie	Sleeping, no activity	TV and DVD are turned ON	Seismic intensity is 5; Remaining time is 10sec	For Person 1, there is not enough time to run to Person 2; An alternative way to help Person 2 needs to be found
6	Reading a book	- (no data)	Room lights are turned ON	Seismic intensity is 7; Remaining time is 3sec	If the policy at the household allows for using the camera, turn it on to possibly assist future rescue operations

Table 4. Pre-earthquake situations

Table 6 shows results – constraint parameters obtained as well as main actions undertaken – of the electronics control by the system. An example of the calculation of achievement degrees in Situation 1 is listed below:

a_1: the number of appliances actually turned off is 10 / the total number of the appliances in the room is 13,

a_2: safety level 5 is obtained / safety level 5 is requested,

a_3, a_5 and a_6 are always assigned by the constraint,

a_4, a_7, a_8 are not calculated because c_4, c_7, c_8 are not sent by countermeasure agents (indicated with "-").

Note that in Situation 5, constraint c_4 cannot be completely satisfied, since the remaining time is too short (10c). The system then issues an alternative recommendation "Make a contact (via a mobile phone)" with a safety level 2 through solving the optimization problem defined with constraints c_2 and c_4.

Constraint number	Constraint specification (i=1,…,n)	Agent	Constraint weight	Achievement degree	Explanation
c_1	state(ce_i, all electronics), ce_i = "OFF"	ECA	3	Electronics actually turned off / all electronics in the room	Turn off electronics for preventing electric shocks
c_2	act_h = "Hide under furniture", h: adult	ECA	5	Depends on the human action's safety level	Act to maintain own safety
c_3	state(ce_i, Heater), ce_i = "OFF"	ECA	6	This constraint must always be satisfied	Turn off all the equipment, which may cause fires
c_4	act_h = "Accompany a", a:with "no activity", h:adult	PCA	5	Depends on the human action's safety level	If there is any "weak" person (e.g. child), act to help her or him
c_5	state(ce_i light), ce_i = "On", (ce_i located near the person)	PCA	4	This constraint must always be satisfied	To facilitate human evacuation and record the person's location for rescue operations
c_6	state(ce_i, Speaker), ce_i = "Announce(earthquake information)"	PCA	4	This constraint must always be satisfied	Deliver earthquake information and navigate the person
c_7	state(ce_i, Speaker), ce_i = "VolUp"	PCA	4	This constraint must always be satisfied	System can do "VolUp" and "Announce(X)"at the same time
c_8	state(ce_i, Camera), ce_i = "Record"	PCCA	2	Depends on the household policy settings	Make a video record for rescue or future analysis

Table 5. Constraint list for Room 1 agent

5. Simulation experiments

5.1 Simulator

In the previous section, we described constraints utilized to control consumer electronics, as well as instructions apparently suitable to guide the evacuation process. Generally however, there is always a chance that the people (and, to a less extent, the equipment) would not act as expected. Another possible source of complications is the fact that people would almost unavoidably interact with each other when evacuating.

Situation number	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	System actions
1	77%	100%	100%	-	100%	100%	-	-	Issue a warning about the earthquake with speakers. Advise to hide under the table. Turn on lights on the evacuation route. Unnecessary equipment is turned off.
2	77%	100%	100%	-	100%	100%	100%	-	Wake up the person, turn on the light in the room, and issue a warning about the earthquake; advise to hide under the table. Turn off unnecessary equipment.
3	77%	83%	100%	-	100%	100%	100%	-	Wake up the person, turn on the light in the room, and issue a warning about the earthquake; advise to get under the bed (to stay away from dangerous objects). Turn off unnecessary equipment.
4	85%	100%	100%	100 %	100%	100%	-	-	Announce the earthquake information with speakers. Advise to go to Room 2 to help Person 2; advise to stay away from dangerous objects. Turn on lights on the route to Room 2.
5	70%	100%	100%	67%	100%	100%	-	-	Announce the earthquake information with speakers. Advise to contact Person 2 via a mobile phone. Turn off unnecessary electronics.
6	77%	100%	100%	-	100%	100%	-	100%	Announce the earthquake information with speakers. Start recording with the web camera.

Table 6. Constraint achievement degrees and system actions

To explore these factors, we have conducted experiments for estimating human evacuation rates with external guidance by a prototype of the proposed system (results of one experiment comparing evacuation rates for a house with and without external guidance by the developed prototype were reported elsewhere (Kokawa et al., 2007); Section 5.3 presents a similar experiment, in which the evacuation rates were estimated for a public facility). Since it is usually impractical to test systems with goals analogous to the ones of the developed prototype in real-world settings on a sufficiently (for validation purposes) large scale due to high risks (for both involved people and equipment) associated with the evacuation process, we have developed a simulator capable of reproducing living spaces (e.g. houses, apartments, and the like) and public facilities (e.g. schools, hospitals, cinemas, etc.) with people in there.

The developed simulator allows for evaluating possible human behavior under external guidance, and it calculates the evacuation rate for different guidance policies emerged from instructions issued by the induced disaster prevention system prototype. The simulator thus operates with the household status database and can dynamically change the simulated evacuation policy.

With the simulator, people are represented as "actors" having 4 parameters: a reaction time, a (running) speed, chances of recovery after a collision, and a recovery time. The reaction time (RT) is the delay from the moment when evacuation instructions are issued until when the actor reacts (e.g. starts moving). This time is randomly assigned, based on a lognormal RT model obtained empirically from data of the real human RT to the prototype guidance (an outline of the corresponding experiment is given in the next paragraph). The running speed is the average moving speed, with which actors would proceed towards an exit (the speed is assigned randomly, based on a Gaussian probability distribution model). Chances of recovery after a collision and the recovery time for an actor are determined by utilizing statistics of evacuation processes reported in the literature (D. Heilbing, 2000).

People normally need time to think and decide upon their actions. Therefore, when evacuation instructions are compiled, a realistic human reaction time should be taken into consideration by the system. Fig. 3 shows results of the experiment conducted to obtain a human reaction time model. The model is to create a random RT generator for the simulator. Two types of subjects have been involved in the experiment: in the figure, "Priming" stands for a group of 30 subjects, who knew about the experiment in advance (the solid line), "No priming" – for 30 (15+15) subjects, who did not possess knowledge about the planned evacuation (the dashed and dotted lines). It is well known that the variance (σ^2) and the average (μ) of human reaction time may differ significantly, depending on the individual's prior knowledge (R. D. Luce, 1986).

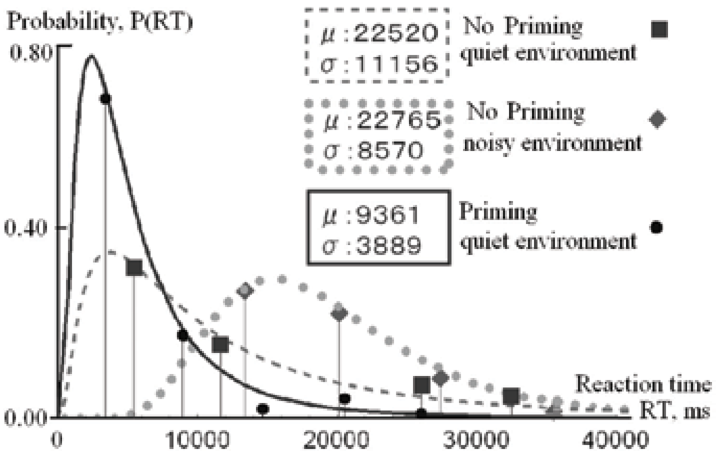


Fig. 3. Human RT patterns in the evacuation process (μ and σ are the parameters of the lognormal MLE fit to the data; the curves thus show the "best-fit models" obtained from the experimental data)

Two types of situations were explored for the "No priming" group: a quiet, and a comparatively noisy, distracting environments. As can be seen from the figure, the

"Priming" group and the "No priming, quiet environment" group demonstrated, though expectedly, a significantly faster reaction, on average.
Actors' behavior in the simulation is defined with a few simple rules, which are listed below:

- Actors cannot walk through each other, walls, and furniture.
- Actors start acting (e.g. moving) after a delay determined by the reaction time.
- Actors try to avoid collisions if there is a physical object (e.g. another actor) ahead.
- If an actor collides, it will be delayed or, by chance, even permanently stopped ("killed" in a trample).
- Actors will be delayed when passing in front of an opening door or going on stairs.
- Actors have prior knowledge about the physical environment and the evacuation paths.

5.2 Experiment 1

Fig. 4 shows the living space reproduced in the simulation. Room 1 of the space is the room used in the case study described in Section 4. Totally, there are 3 "big" rooms with the corresponding room agents installed. These room agents are also "in charge of" the adjacent spaces, such as corridors, closet, kitchen, etc. A family of three – "father," "mother," and "child" – was modeled in the experiment as follows.
Father: is fully aware of the installed system and its capabilities; in day-time, is usually not at home; is usually "associated" with a quiet environment.
Mother: is aware of the installed system; spends a significant part of her time at home; in day-time, is mainly "associated" with a noisy, distracting environment (e.g. due to housekeeping activities).
Child: does not know about the installed system; needs help when an earthquake occurs.

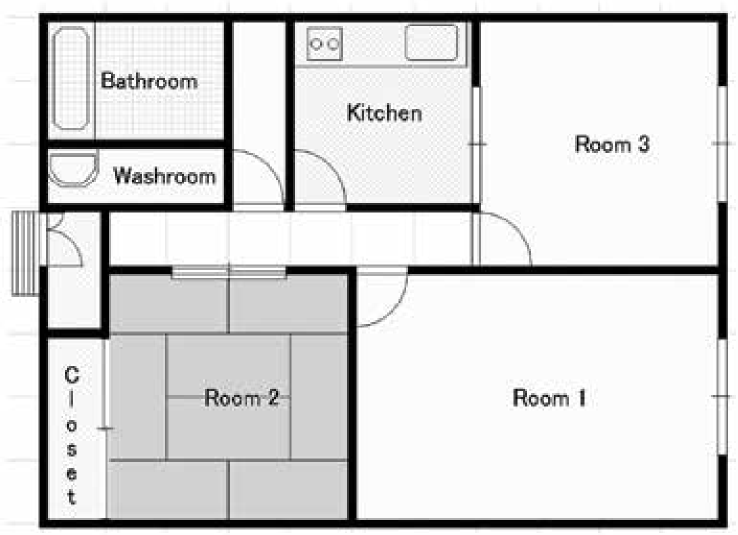


Fig. 4. Living space used in Experiment 1

The flow of the simulated events is as follows. First, an earthquake early warning is initiated at random, and the corresponding data – the seismic intensity and the remaining time – are randomly set. The room agents then decide upon the control of consumer electronics specified in the simulator as "installed equipment," and the appropriate evacuation strategies (if any) are implemented. Finally, the damage caused by the earthquake is calculated.

The statuses of the family members are assigned based on the time of the simulated earthquake, as well as on the "most typical/expected" behavior of the members, who are "at home" at the given time. Human responses to the external guidance are modeled, using the detailed information of each room's layout (e.g. see Fig. 2 for the details on Room 1), the relevant RT model, and the current statuses of the inhabitants. The damage *D* caused by the earthquake is calculated as follows:

$$D = \sum_{n=1}^{NQ} \left(\alpha \sum_{i=1}^{people} \begin{cases} 0, if\,\text{�}\,level < dp_i \\ 1, if\,\text{�}\,level \geq dp_i \end{cases} + \beta \sum_{j=1}^{areas} \begin{cases} 0, if\,\text{�}\,level < dt_j \\ 1, if\,\text{�}\,level \geq dt_j \end{cases} \right). \tag{2}$$

As it may be understood from formula (2), the damage is estimated for *NQ* simulated earthquakes, and it includes the "human damage" (summation over the people present at the time of the earthquake) and the "living space damage" (summation over the living space areas); *level* is a simulation parameter proportional to the strength of the simulated earthquake, dp_i and dt_j are thresholds assigned from empirical data reported in the specialized literature (Heilbing, 2000), and α and β are coefficients representing the corresponding damage rates, which are set to values given in post-earthquake reports by the governmental organizations (Government of Japan, 2006).

In this simulation experiment, 3 different disaster-prevention strategies were evaluated: when earthquake information is simply announced with the available electronics (the case of the JMA early warning system – see Section 1), when earthquake information is announced and simple disaster-prevention countermeasures are executed (the case of the JEITA system), and when the full range of the prevention measures available to the system is duly executed (the case of the proposed system). Fig. 5 shows the simulation results. As it

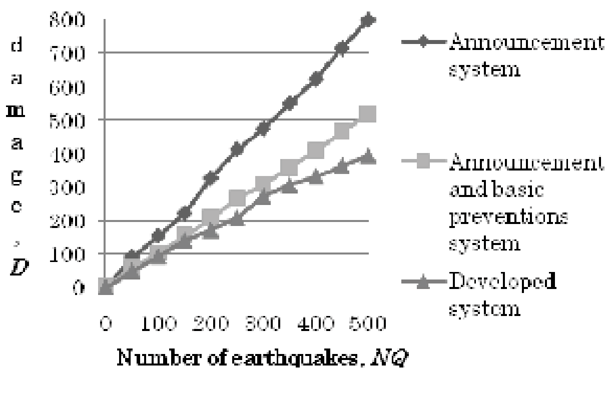
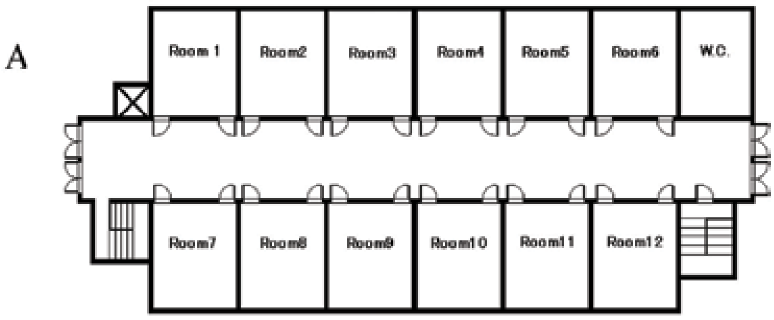


Fig. 5. Experiment 1: Simulation results

can clearly be seen from the graphs, the simple disaster-prevention strategies (lines with filled diamonds and squares), even though appear useful from a common-sense point of view, provide for a significantly lower level of safety (i.e. greater earthquake-induced damage) than the more sophisticated, adaptive strategy implemented with the system developed in this study (shown with filled triangles) does. Under other similar conditions, the developed system would help reduce, in a long run and on average, the earthquake-induced damages by nearly a half, compared to the simplest case of the JMA early warning system.

5.3 Experiment 2

We have also conducted a simulation experiment to estimate the possible effect of the guided evacuation of people from a public facility on the evacuation rate. A university's 3-story building was modelled with the simulator. This building is normally full of students (the estimated student number is 500÷1000) in class-hours. Fig. 6 shows the layout of the building, which has 12 classrooms in its 1st floor and 6 larger lecture halls in each of its higher floors; the floors are connected with two stairs at both sides of the building, where the exits are located. In the simulation model, we did not include the W.Cs and the elevator (denoted X in the figure).



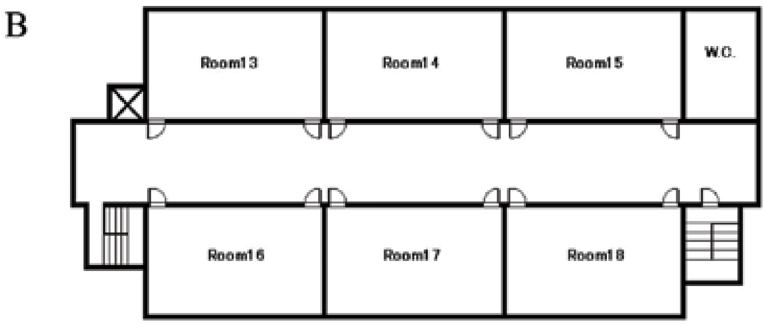


Fig. 6. Layout of the simulated public facility: A - 1st floor, B – 2nd and 3rd floors

In the model used, a teacher was positioned in front of every classroom, and students were randomly distributed over the seats available; the number of students in every class was randomly assigned in the range from 25 to 75% of the full capacity. It was assumed that when an announcement is made (or an alarm is activated), people inside of the rooms will start moving after a delay, as it is stipulated in Fig. 3 for the "Priming" case. The people will then try to get out of the building, using the relatively narrow doors and stairs. A 3D model of the building with people evacuating is shown in Fig. 7, where the cylinders represent people at a time in the middle of evacuation.

In the simulation experiment, two types of the evacuation guidance strategies were compared. In the first case, it was assumed that the people, all together, start evacuation when a simple alarm is activated in each room (Sim1 in Fig. 8). The second strategy implies the use of the earthquake-induced disaster prevention system with the room agents installed in the classes. The developed system prototype then decides on the timing of the announcement and its contents (e.g. "Stay in the class," "Quickly move outside! Go to the front stairs," etc.), using its knowledge base and the monitoring agents. In the experiment shown in Fig. 7, the system recommended for the students in rooms 1, 5, 7, 12, 13, 15, 16, and 18 (see Fig. 6) to start evacuation as soon as the earthquake early warning is received; a 5 second delay was recommended for rooms 2, 5, 8, 11, 14, and 17, and a 10 second delay – for rooms 3, 4, 9, and 10. In the given experiment, the total time available for evacuation was varied by the disaster prevention system from 15 to 25 seconds. Results of the evacuation with delays are shown in Fig. 8 as Sim2.

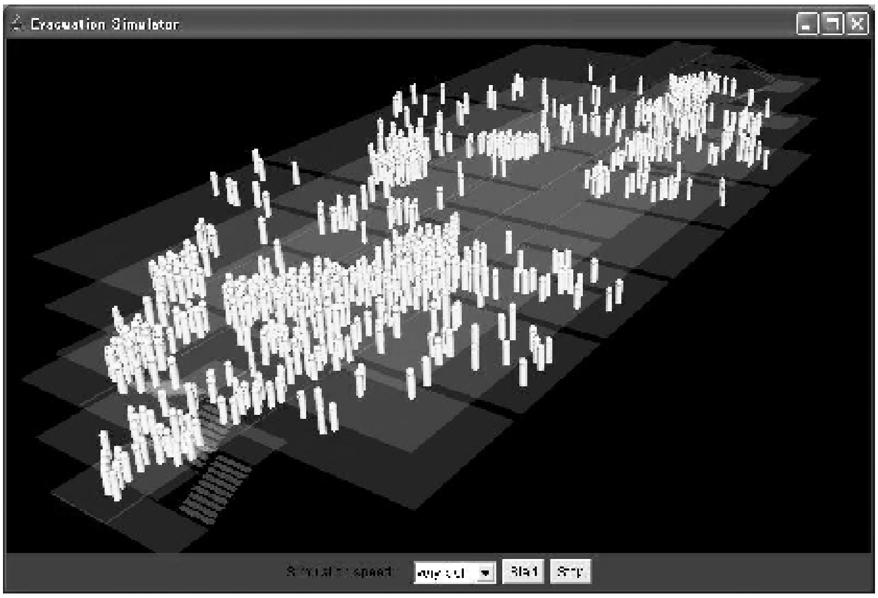


Fig. 7. A screenshot of the simulator during the modelled evacuation process

While the results obtained in the experiment (see Fig. 8, where the graphs are build after averaging over 20 simulations) clearly indicate the advantage of the guided evacuation strategy, it is understood that earthquakes in reality seldom leave us as much as 10 seconds to react. At the same time, however, the purpose of this (second) experiment was not just to test the developed prototype in a different environment and at a larger scale, but rather to show the potential applicability of the proposed system to the case of disasters other than earthquakes – fires, bio- and chemical accidents, etc. The current implementation of the secondary disaster prevention system is quite specialized to deal with the case of earthquakes. It then appears a natural but still unrealized extension of the proposed system concept to generalize it to handle a variety of hazardous situations, possibly using multiple communication networks (in addition to the earthquake early warning network) for obtaining initial data.

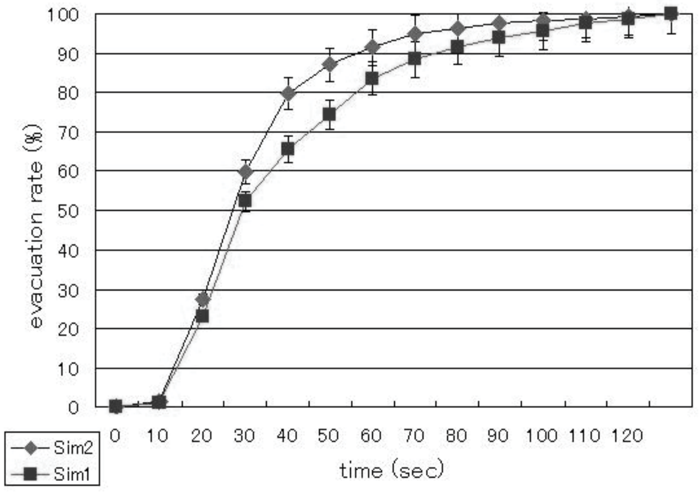


Fig. 8. Results of Experiment 2: the developed system prototype (graph Sim2) provides for a faster evacuation rate than in the case when a simple alarm system, which is usually used at public facilities such as university buildings, is activated (graph Sim1)

6. Related work

Most of the relevant studies reported in the literature deal with earthquake early warning systems to merely deliver earthquake-related information to the inhabitants in an effective way. As a typical example, the Real-time Earthquake Information Consortium system was developed to convey earthquake-related announcements to every home in a particular area using IP phones (REIC, 2004). There also were, however, reports in the past few years about systems that have goals and capabilities similar to the ones pursued in the presented study.

The system proposed by JEITA is an automatic consumer electronics control system (JEITA, 2005). A somewhat analogous system was developed by Seismic Warning Systems Incorporated, using an earthquake early warning network deployed in the West Coast of the USA (SWS, 2004). Another relevant system was created in Taiwan (Wu et al., 2004). All these systems can control simple electronic devices for the earthquake-induced disaster prevention, utilizing the data obtained from the corresponding early warning systems, e.g. they can shut off gas, issue warnings, open door locks, and so on. The systems have, however, to have countermeasures defined for every possible scenario in advance and, hence, if the environment changes, the recommended actions may become ineffective or even dangerous. Besides, the consistency of the systems' knowledge bases appears hard to maintain due to the changing surroundings.

A wide area of research, which is closely related to our study, is the creation of global and local telecommunication infrastructures (networks) that could be used by various disaster-prevention systems. Although not explored in detail in our experiments, it may be natural to expect that the robustness and reliability of the corresponding data- and information-networks will, to a large extent, determine the efficiency of automated disaster-prevention systems. The agent-based design proposed in our study can utilize, in its agent-to-agent-communication part, the best solutions reported in the literature (Harayama & Inoue, 2006; Lin et al., 2002).

7. Concluding remarks

In the presented study, an intelligent adaptive system to control consumer electronics and guide the evacuation process based on the earthquake early warning has been proposed. The system has an agent-based architecture, and it dynamically implements optimized strategies for the prevention of earthquake-induced disasters by solving an achievement-weighted constraint satisfaction problem. The system's sensor-based latent interface allows for adjusting disaster prevention control policies, depending on human behavior in (pre-) earthquake conditions. The system is thus able to adapt to dynamic environments, as its room agents monitor the populated space and update the system's knowledge- and data-bases.

A system prototype has been created and used in a case study and tests conducted with a simulator, which has also been developed in this research. Experimental results obtained demonstrated that the proposed design solutions provide for a significantly higher level of safety for people in hazardous situations, when compared with the existing disaster-prevention systems.

In future work, we plan to increase the flexibility of the system by diversifying control and evacuation strategies potentially available, to connect the system agents to a consumer electronics network, and to develop a new version of the system prototype capable of operation in realistic earthquake conditions. An augmentation of the simulator functionality to provide for dealing with more environmental parameters and to diversify the possible (simulated) interactions among the actors is also planned. A more challenging task remains to develop a secure, efficient and effective but inexpensive technology to monitor human behavior and update the dynamic status database.

8. References

S. Bistarelli, P. Codognet, F. Rossi (1999), Abstracting Soft Constraints: Framework, properties, examples, Artificial Intelligence, Vol.139, 1999, pp. 175-211

K. Doi, (2002). Earthquake early warning system in Japan, In: *Early Warning System for Natural Disaster Reduction*, Jochen Zschau, (Ed.), Springer, Berlin

Government of Japan, (2006) Earthquake occurred in Japan, Disaster Prevention Information Page. Abstract of earthquake countermeasures in Japan, Available from: http://www.bousai.go.jp/jishin/chubou/taisaku_gaiyou/pdf/hassei-jishin.pdf
(last accessed on May 04, 2008) (In Japanese)

H. Harayama, M. Inoue (2006). Study on home network for realization of real-time disasters prevention system, *Information Processing Society of Japan*, Vol.2006, No.54, 2006, pp. 39-42, ISSN: 0919-6072 (In Japanese)

D. Heilbing (2000). Simulating Dynamical Features of Escape Panic, *Nature*, Vol. 407, Sep 2000,. pp. 487-490

JEITA (2005). IT automatic disaster prevention system, Second Press Release, *Japan Electronics and Information Technology Industries Association (JEITA)*, Japan, 2005.12.7 Available from: http://home.jeita.or.jp/spp/index.html (last accessed on May 04, 2008) (In Japanese)

A. N. Kueppers (2002). Early Warning System for Natural Disaster Reduction, Springer, Berlin

M. Kikuchi (2004). The development of a software system to utilize the earthquake real-time information, *Ministry of Education, Culture, Sports, Science and Technology Research Report*, pp. 410-425, Tokyo ,Japan, (In Japanese)

T. Kokawa, H. Ogawa (2004). A decision-making support system in consideration of individual preferences, *The Institute of Electronics, Information and Communication Engineers(IEICE)*, Technical Report, Vol. 104, No. 133, Tokyo, Japan, 2004, pp. 1-6 (In Japanese)

T. Kokawa, Y. Takeuchi, R. Sakamoto, H. Ogawa, V. V. Kryssanov (2007). An Agent-Based System for the Prevention of Earthquake-Induced Disasters, *Proceedings of IEEE International Conference on Tools with Artificial Intelligence*, Oct. 2007, pp. 55-62, ISBN: 978-0-7695-3015-4

Y. Lin, H.A Latchman, M. Lee, and S. Katar (2002). "A power line Communication Network Infrastructure for the Smart Home", *IEEE Wireless Communications*, Vol.9, Issue.6, 2002, pp. 104-111, ISSN: 1536-1284

R.D. Luce (1986). Response times: their role in inferring elementary mental organization, *Oxford University Press*, New York, USA

X. Luo, J. H. Lee, H. Leung, N. R. Jenningsa (2002), Prioritised fuzzy constraint satisfaction problems: axioms, instantiation and validation, Fuzzy Sets and Systems, Vol. 136, No. 2, 2002, pp. 151-188

REIC (2004) Research and Development of Automatic Disaster Prevention System towards Domestic Controlled Network (IP Phone), *Real-time Earthquake information Consortium* (REIC) http://www.real-time.jp/research/lp/lp-04b.html

T. Schiex, H. Fargier, G. Verfaillie (1995), Valued constraint satisfaction problems: Hard and easy problems, In Proc. IJCAI-95, 1995, pp. 631-637

SWS (2004), Corporate Backgrounder, *Seismic Warning Systems Inc. (SWS)*, Corporate Overview, 2004

M. Walliser, S. Branschen (2004). M. Calisti, T. Hempfling, Constraint Satisfaction Techniques for Agent-Based Reasoning, Brinkhauser Verlag, Berlin, 2004

Y. M. Wu, T. L. Teng, N. C. Hsiao, T. C. Shin, W. H. K. Lee and Y. B. Tsai (2004). Progress on Earthquake Rapid Reporting and Early Warning Systems in Taiwan, In: Y. T. Chen, G. F. Panza, Z. L. Wu (Ed), Earthquake Hazard, Risk, and Strong Ground Motion, *Seismological Press*, Berlin, 2004, pp. 463-486

M. Yokoo (2001), Distributed Constraint Satisfaction, *Foundations of Cooperation in Multi-agent System*, Springer-Verlag, New York

25

A Methodology for the Extraction of Reader's Emotional State Triggered from Text Typography

Dimitrios Tsonos and Georgios Kouroupetroglou
National and Kapodistrian University of Athens
Greece

1. Introduction

Writing is employed by humans in order to communicate, exchange ideas or store facts and descriptions. A well known Latin phrase is "*verba volant, scripta manent*" i.e. "spoken words fly away, written words remain". Humans in prehistoric years, used figurations to describe several events and express their fear or their admiration, the ancient Egyptians used papyrus to write down their ideas and the Chinese, in 11th Century, went beyond handwriting using typography with moveable type to create multiple copies of their documents. Through centuries typography has evolved and nowadays is one of the major manners to exchange ideas and information.

Using computers, typography and the way we write has changed radically. We have a plethora of software tools, e.g. for writing a plain text (like NotePad in MS Windows), word processors (e.g. Open Office) for text editing as well as for more elegant visual appearance of the documents and professional tools for the creation of posters, advertisements e.t.c. (e.g. Adobe's Tools and Editors). Utilising these tools, computer users have the ability not only to write a plain text, but also to format the text and arrange it in the page. In newspapers, for example, the page designer distinguishes the title from the body of text at the top of the page or the article, with larger font size. Also, when an editor wishes to emphasize a specific word or phrase he uses bold or italics typesetting. A writer can convey a message, a felling or an idea not only by the meaning of the content but also by the way the text is visually presented to the reader. Page layout affects the way a newspaper is read (Holmqvist & Wartenberg, 2005; Holmberg, 2004; Küpper, 1989; Wartenberg & Holmqvist, 2005).

The use of the WWW and the web page creation and design, introduced a new perception for the meaning of documents and publishing. In web pages, the text and background color combinations have impact on the readability and aesthetics (Porat et al., 2007; Richard & Patrick, 2004; Hill & Scharff, 1997) and a well designed graphical web document can be reader friendly (Borchers et al., 1996).

Humans express their emotions in every personal or social occasion. Everything they do or make is followed by or follows an emotional expression. For example, some people are *afraid* of being among many people. So they act accordingly by avoiding congestion. If they cannot, they are feeling *anxious* and *nervous*. Another example of the role of the emotions in

our life is that people find themselves to be more productive when they are *happy* than when being *depressed* or *unhappy*. Thus emotions play a significant role in our daily life.
But, what are emotions? There are theories that are trying to answer this question (James, 1884; Cornelius, 2000; Narayanan & Alwan, 2004) and set additional questions like which is the optimum theory to explain emotions and how can those be measured (Scherer, 2005).
This chapter aims to present how emotions can be used in Artificial Intelligence, for the automated extraction of the reader's emotional states. Starting with a survey on the basic theories on emotions, a sort description of the existing experimental procedures for measuring emotions follows. We present the Self Assessment Manikin Test (Lang, 1985) as a tool for modelling reader's emotions and emotional states such as "Pleasure", "Arousal" and "Dominance" (Tsonos et al., 2008). In the second part of the chapter there is a sort description of what typography is, what typographic elements are used in documents and how they affect the reader emotionally. In the third part we propose a system architecture for the automated extraction of readers' emotional state along with the necessary steps that should be followed during its implementation.

2. Emotions and emotional states

Through the years, several studies and theories on emotions have emerged. In this section, we describe four theories - approaches on emotions. The theoretical survey helps to understand and investigate how reader's emotions and emotional states are affected by text typography, how these emotions can be measured and the optimum experimental procedure that can be used for modelling emotions.

2.1 Theories on emotions and emotional states

Emotional theories are classified as:

- the Darwinian Theory,
- the Jamesian Theory,
- the Cognitive Theory,
- the Social Constructivist Theory.

The Darwinian Theory assumes that emotions "are evolved phenomena with important survival functions that have been selected for because they have solved certain problems we have faced as species" (Cornelius, 2000). All organisms sharing an evolutionary past should share the same emotions as well. The emotions should be analyzed in terms of their functionality and survival value (Cornelius, 2000; Narayanan & Alwan, 2004).
According to the Jamesian Theory, "bodily changes follow directly the *perception* of the exciting fact and our feeling of the same changes as they occur *is* the emotion" (James, 1884). James's belief on emotions is that, we experience emotions because our bodies have evolved to respond automatically and adaptively to features of the environment that have survival-related significance to us. Our bodies respond first and our experience of these changes constitutes what we call emotion (Cornelius, 2000). In this approach, if we consider a system that is a "black box", the stimulus (e.g. document, speech) is considered to be the input of the system while the output is the bodily change. Thus, we can change someone's emotions as we like just by changing the stimulus (Narayanan & Alwan, 2004).
The Cognitive Theory proposes that "every emotion has associated with it a particular pattern of appraisal is that if the appraisal is changed, the emotion should change as well" (Cornelius, 2000). The thought and emotions cannot be separated, and what one feels

depends in how one understands and judges the events. Emotions result from mental activity (Cornelius, 2000; Narayanan & Alwan, 2004).
Finally, according to the Social Constructivist Theory, culture and social rules play a key role in the organization of emotions at a variety of levels. The last two theories are closely related, and the development of thoughts is tightly connected to culture and social status (Narayanan & Alwan, 2004).

2.2 Experimental procedures for emotional state extraction

According to Scherer, emotion is defined as "an episode of interrelated, synchronized changes in the states of all or most of the five organismic subsystems in response to the evaluation of an external or internal stimulus event as relevant to major concerns of the organism" (Scherer, 2005).
Psychologists have developed several tools and experimental procedures that can measure emotions. They can be categorized into Free Response Measurement and Forced Choice Response Measurement.
Using Free Response Measurement, the participants of an experimental procedure are asked to respond with freely chosen labels or short expressions that characterize the nature of the emotional state they experienced. Using this procedure, instructors and researchers may face problems, like in case where the participants may not be able to express their emotions due to the use of inappropriate labels or the limited range of their vocabulary that may constrain their responses. Also, it is quite difficult to statistically analyze free responses (Scherer, 2005). Researchers sort numerous free responses into a small number of emotion categories using synonyms and resemblances. It is not a standard experimental procedure but efforts have been made towards standardizing the emotional labels.
Forced Choice Response Measurement can be subdivided (Scherer, 2005) into a) Discrete Emotion approach and b) Emotion Dimensions approach.
In Discrete Emotion approach, the participants are asked to assess their emotions using verbal expressions that best describe their emotions or provide feedback on a 3 to 5-point scale indicating whether the emotion experienced was weak or strong or use an analog scale to indicate the intensity of an experienced emotion.
There is a standardized measurement procedure for this type of experiments, but many researchers prefer to develop their own emotional categories. Such approaches result to mismatched categories that may present problems concerning the comparability of the results (Scherer, 2005). There are also problems in cross-study statistical analyses due to the abundance of missing data.
The participants in Emotion Dimensions procedure, are asked to denote how positive (pleasant) or negative (unpleasant) and how aroused (excited) or calm they feel. The emotions can be mapped using the bi-dimensional space of Pleasure-Arousal. This method is very simple, straightforward and reliable (Scherer, 2005). Also, simple or advanced single study or cross-study statistical analyses can be obtained in contrast with the Free Response Measurement and Discrete Emotion approach. In this case it is difficult to differentiate the intensity of the feeling from body excitation (Scherer, 2005).

3. Emotional state assessment using the Self-Assessment Manikin test

3.1 The S.A.M. test

The Self Assessment Manikin (SAM) Test was introduced in 1985 by P.J. Lang (Lang, 1985). The SAM procedure came up as a test for the assessment of advertisements (Bagozzi et al., 1999). It offers the ability to avoid the verbal expression of the emotion assessment, so it establishes a quick and easy to use experimental procedure. Also, having a pictorial assessment rather than a verbal one makes SAM test cross–cultural and language-independent (Morris, 1995).

The test assesses the *emotional states* of the participants. These states are "Pleasure", "Arousal" and "Dominance" (The SAM test is also known as "PAD test" the initials of each emotional state). Synonyms are used for the expression of the PAD dimensions. "Pleasure" can be replaced by "valence" and "evaluation", "Arousal" by "activation" and "activity" and "Dominance" by "power" and "potency". In our study we used the initial verbal expressions of the three dimensions.

Using the three-dimensional space of emotional states, we are able to map them into specific emotions. A well-known example is the Russell's circumplex (Scherer, 2005). The two dimensions of "Pleasure" and "Arousal" are represented on a X-Y grid respectively. Russell placed the verbal expressions of the emotions on the grid. Figure 1 illustrates the emotional wheel with the verbal semantic mapping of the emotions.

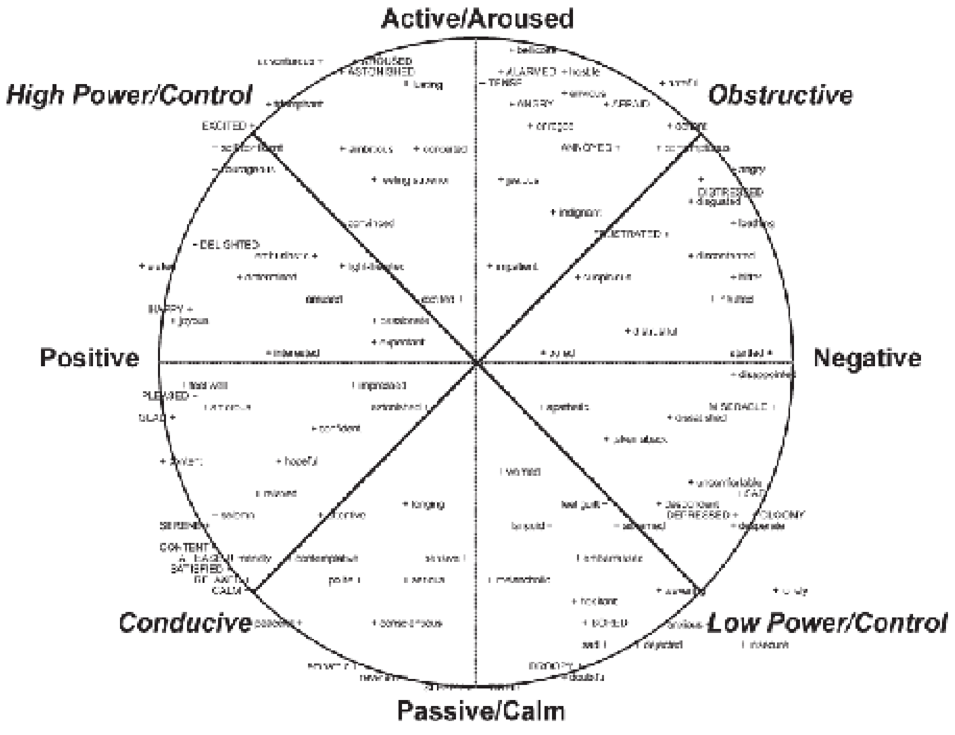


Fig. 1. The verbal expressions of the emotions placed on the Pleasure–Arousal grid according to Russell

Another version of emotional wheel is the Geneva Emotion Wheel (GEW) introduced as an experimental tool (Scherer, 2005; Bänziger et al., 2005). A computer screen of the GEW is presented in Figure 2.

How do we assess the emotional states and how do these values derive and can be presented on an X-Y grid? The SAM test, as it is described, can "produce" the values of the Pleasure, Arousal and Dominance dimensions.

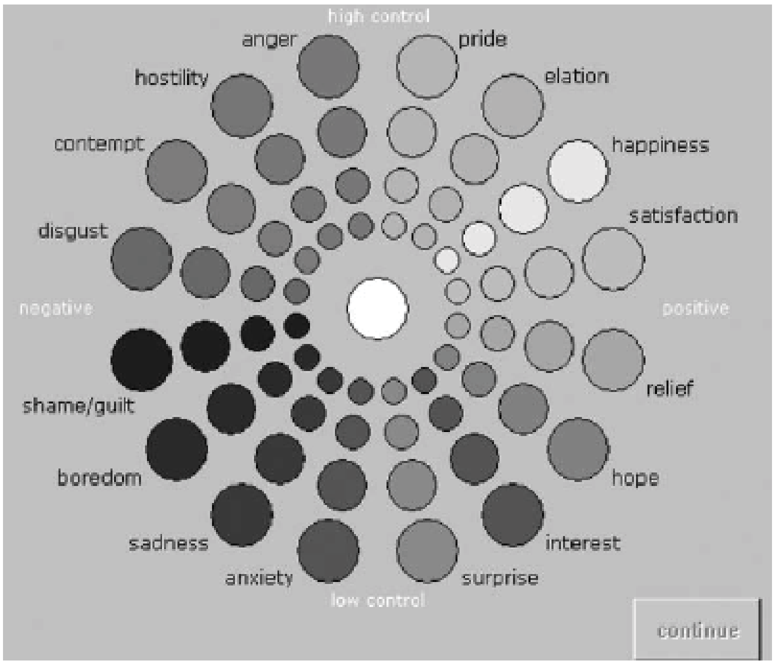


Fig. 2. A computer screen of the Geneva Emotion Wheel

During the experimental procedure, the participants assess their emotional states using five manikins (example in Figure 3) for each dimension of Pleasure, Arousal and Dominance.

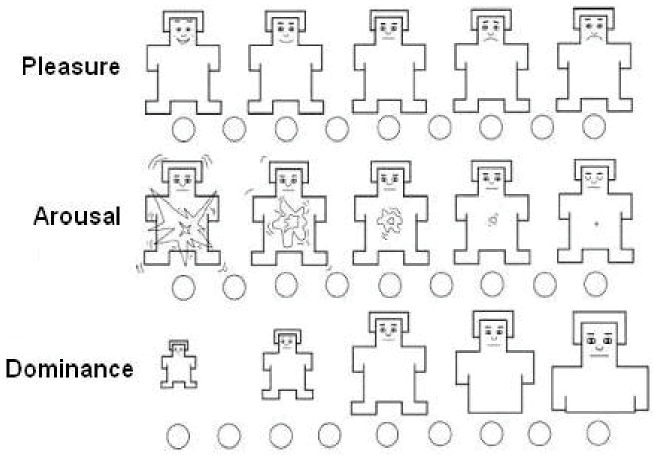


Fig. 3. The manikins of the 9-point scale SAM Test as presented during the experimental procedure. The verbal expressions of "Pleasure", "Arousal" and "Dominance" do not appear during the experimental procedure

There are no verbal expressions to assess their emotional states. The participants can choose from at least 5 selections of manikins. In some studies (Morris, 1995; Tsonos et al., 2008) are presented by larger scales of 9-points to 25-points. For example, in a 9-point scale there are 5 points of selection for the images and 4 points as interval values. The choice of the scale depends on the design of the experimental procedure and the number of the participants. If there is a need for more accuracy for the results a 9-point scale or greater should be selected. The scale of the experimental procedure is proportional to the number of subjects needed for acceptable statistical results.

For the emotional state of "Pleasure", the rating begins from a happy, smiling manikin to an unhappy, frowning one. For the "Arousal" dimension the aroused pole is represented by a highly energetic manikin and the other pole is represented by a relaxed and eyes-closed one, while for "Dominance" the controlled and in-control poles are represented by a small and large manikin respectively.

The answers of the participants can be transformed from the 1-5 point scale (or 1-9 point scale, etc.) into a dimensional space of [-1,1] or [-100%,100%]. Using the percentage approach we are able to distinguish how much an emotional state has been varied from the neutral state (the value "zero" represents the neutral state in both spaces).

3.2 Using SAM test as an experimental tool in artificial intelligence: case studies

Using SAM test we can have two different results that are closely related, the dimensional perspective of emotions and the emotions. Using only one experimental procedure, the researcher is able to have two approaches on his research.

Conducting this kind of test, modeling the emotions and emotional states of the participants, on specific projected stimuli, is feasible.

In (Grimm et al., 2006; Grimm et al., 2007a; Grimm et al., 2007b) a system was implemented for the automated estimation of emotion primitives (the dimensional approach) from speech using acoustic features.

There is also a study on the automated detection of pleasant and unpleasant emotions in spoken dialogs (Chul & Narayanan, 2005) obtained from a call center application.

Busso (Busso et al., 2007) modeled head motion sequences in expressive facial animations, analyzing them in terms of their naturalness and emotional salience in perception (SAM test is used for the evaluation of the results).

3.3 Description of the experimental procedure

In this paragraph we present a version of the SAM experimental procedure, designed for our experimental needs, following the paper and pencil IAPS Guidelines (Lang et al., 2005). The automated procedure helps to create an easy of use experiment and the rapid collection and process of the results.

For the development of the procedure, PHP was used on an Apache Web Server and MySQL. PHP allowed us to develop dynamic web pages, for the presentation of the stimuli, and to automate the registration of the participants' answers. The answers were stored in a database (MySQL). Before the experimental procedure began, participants read a short description of the purpose of our study and explicitly tutored in the emotional states theory. Certain guidelines were followed for all participants during the experimental procedure (Lang 2005). They were given ample time to read the instructions and freely ask the instructors for any additional information or clarification. Also, they were asked to complete an electronic form with some personal information, for example about their age, education level, for any visual problems, and also if she/he agrees to participate in the experimental procedure.

The participants were familiarized with the test before the actual experimental procedure. Each stimulus was presented for a few seconds after which the participant was asked to complete the form with the manikins as presented in Figure 3. By pressing the "continue" button, she/he was presented with the next stimulus and so on. The presentation of the stimuli was in the same random sequence to all the participants.

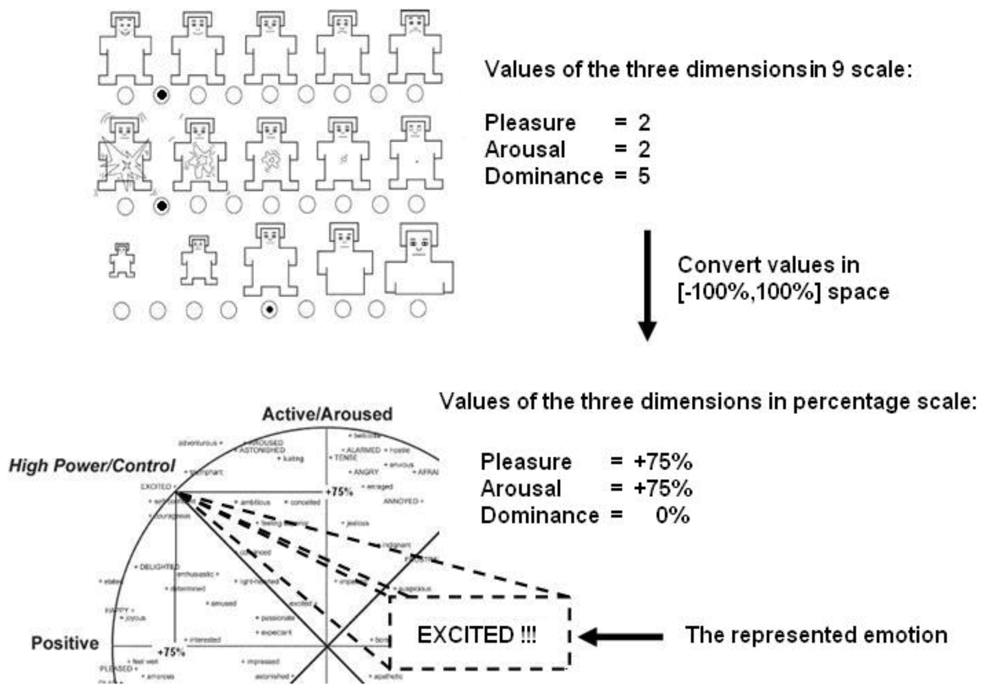


Fig. 4. Extracting the emotion using the SAM Test. Using the manikins, participants assess their emotional state. The 9-point scale values are converted into a percentage scale and then mapped on to the emotion wheel for the verbal-semantic representation of the emotion

4. Text typography in documents

The origin of the word "Typography" comes from the Greek word "Τυπογραφία" which is derived from the words "τὺπος" (″τὺπτω″ = to strike) and "γράφω" (=to write). Through centuries, typography has evolved using different techniques. In 20th century, Computer and Information Science created a new framework for the typography. Using computers, the layout designers of books, magazines etc. can create good aesthetic and functional results on documents in a very short time.

But in the last few years the concept "typography" has been changed due to the wide spread of computers and internet. Most computer users and readers prefer to read documents from their computer screens. The classic printed documents are transformed into electronic documents. Readers are given the possibility and functionality to interact with content of electronic documents. The reader can search specific sentences or phrases in the document, request for a summarization of specific parts of the document and browse a collection of documents.

In this section we will describe how the typographic elements can be categorized and how these elements or their combination affect the readability of the document.

4.1 Classification and usage of typographic elements

There are many efforts that categorize and classify the typographic elements, either in printed or electronic documents, according to the needs of each study. W3C (W3C, 2008a), DAISY/NISO (DAISY, 2008; ANSI/NISO, 2008) standard and Open Document Format (ODF) by (OASIS, 2008) are the three major contributors towards that classification.

In our study, the elements are categorized using the guidelines provided by these standards. The typographic elements of the documents will be mentioned as meta-data. These meta-data can be categorized in (Tsonos et al., 2007a):

- Text Formatting,
- Text Structure,
- Text Layout,
- Non-textual (Figures, Drawing, Pictures, Logos etc).

Text Formatting meta-data include the formation elements of the text, typesetting elements and font elements (like bold, italics, font size). Text Structure meta-data specify the attribute of a part of the document (chapter, title, paragraph etc.), while the Text Layout meta-data describes the visual layout of the text (like columns, headlines, borders). The text with the formation and structural metadata can be combined with other non-textual metadata, such as figures, drawing etc.

One can notice that there is a relation between these elements. For example, a title (text structure) element may have a 16pt font size (text formatting) and placed in a text column (text layout). A subtitle element may have 14pt font size and placed in a column, but also under the title.

Often, textual features present purely logical structure information. For example, a piece of text can be part of a list or footnote. However, frequently, the use of these features has the purpose of communicating semantic information to the reader. Examples of such purposes are: giving emphasis to an important piece of text; focusing the readers' attention to the central point of an article; highlighting a name entity like the title of a movie and so on. This classification can help the creation of sophisticated document manipulation tools including an enhanced audio representation of the document.

Recently there was an attempt to produce an automatic extraction system of semantic information based only on the document layout, without the use of natural language processing (Fourli-Kartsouni et al., 2007). However, there are several studies on the automatic identification of logical structure of documents e.g. (Conway, 1993; Yamashita et al., 1991; Derrien-Peden, 1991; Krishnamoorthy, 1993). Most traditional approaches in this field have employed deterministic methods (decision trees, formal grammars) (Mao et al., 2003; Tsujimoto & Asada, 1990; Derrien-Peden, 1991), which may suffer from poor performance due to noise and uncertainty. In addition, such approaches create models which are not flexible to domain changes and cannot easily evolve in the presence of new evidence.

In order to overcome such limitations, Fourli-Kartsouni (Fourli-Kartsouni et al., 2007) employed a probabilistic approach based on Bayesian networks trained on a series of labelled documents. Bayesian networks offer a significant tolerance to noise and uncertainty, they can capture the underlying class structure residing in the data and they can be trained on examples, thus adapting to existing and future evidence.

4.2 How typographic elements affect reader's emotional state, comprehension and navigation in electronic documents?

E-documents have exceeded documents in printed format by providing several functionalities to the reader like browsing, navigation, searching, highlighting and multimedia facilities. The meta-data and the way it is combined, has specific meaning for each reader in different documents (Küpper, 1989; Holmberg, 2004) and the way the document is read (Holmqvist et al., 2003; Holmqvist & Wartenberg, 2005). The authors utilize structural and layout meta-data to personalize the content. The style of a document sets the typographic rules, used more often in technical or scientific papers and reports than in magazines. There are also studies on how the combination of colors on a page and the different type of style on text, affects the readers' emotional state and the readability of the document (Laarni, 2003), the meaning (Küpper, 1989; Holmberg, 2004) and the reader comprehension.

Emotions and the emotional state of the reader depend on document structure, layout and text formatting. Multiple combinations of colors (Birren, 1984), font size, type and style in a document affects the emotional state (Sánchez et al., 2006; Sánchez et al., 2005) and consequently the readability of the document (Laarni, 2003; Saari et al., 2004) not only in printed but also in electronic format (Larson, 2007).

5. Extraction and modeling reader's emotional state

5.1 Proposed architecture

Based on the SAM test, we proposed an XML-based architecture for the real-time extraction of reader's emotional state *excluding* any content and/or domain dependent information from the input documents (Tsonos et al., 2007b). Figure 5 illustrates a diagram of the system architecture. The proposed system is designed to process all types of documents in printed or electronic format. The Markup Normalization Module converts all non-tagged documents as well as tagged (not conforming to the DAISY/NISO format) into tagged compliant to DAISY/NISO format. Printed documents are scanned and parsed through an OCR system so to be digitized and exported in a tagged format. Documents being already in a tagged format include meta-data about the format and the structure of the text. All these have to be normalized to the required meta-data and file type (DAISY/NISO standard).

The documents, in the desired format, can be processed by the Emotional State Extraction Module. This module is implemented using the model derived by several experimental procedures on how the reader is affected by the document metadata provided in paragraph 4.1. The experimental procedure is similar to one described in 3.3. Conducting multiple experiments using SAM Test we are able to distinguish the way each document element affects the reader's emotional state but also their combination. For the readers' emotional state modelling in paragraph 5.3, the results from ongoing experiments and how the readers' emotional states varying according to font type, size and color, background color and typesetting elements are presented. Future work may reveal the need of a rule based model or a statistical one.

The Emotional State Extraction Module produces an XML file, the Emotional-ML. This file contains the content of the documents but also the initial tagging (after markup normalization) and the emotional state annotation. Currently, there is an effort to create a new markup language by W3C Emotion Markup Language Incubator Group. The group will discuss and propose scientifically valid representations of several aspects of emotional states that appear to be relevant for a number of use cases (W3C, 2008b).

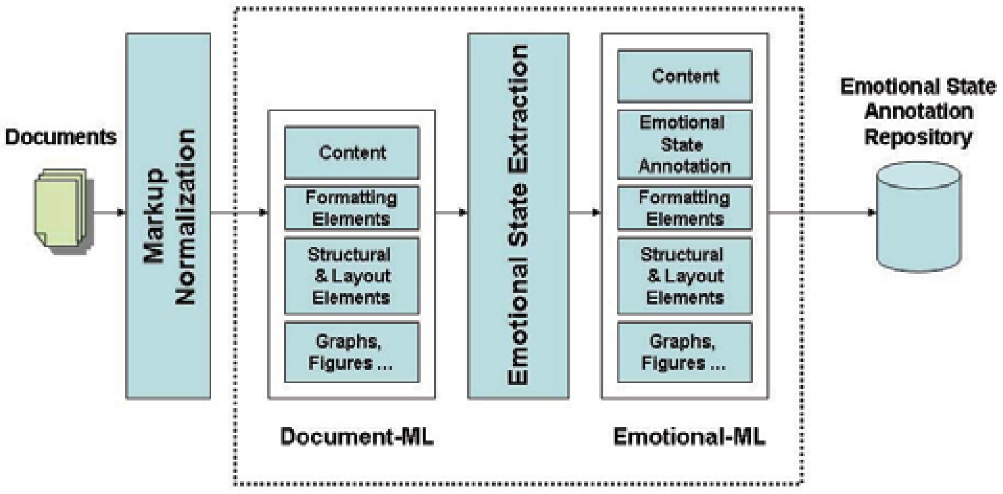

Documents
Markup Normalization
Content
Formatting Elements
Structural & Layout Elements
Graphs, Figures ...
Document-ML
Emotional State Extraction
Content
Emotional State Annotation
Formatting Elements
Structural & Layout Elements
Graphs, Figures ...
Emotional-ML
Emotional State Annotation Repository

Fig. 5. The proposed architecture for the real-time extraction of reader's emotional state

5.2 Experimental methodology

For the reader's emotional state modelling, an experimental procedure has been created according to 3.3, assessing 15 participants. Fifty-four combinations of font and typesetting elements, projected on identical LCD displays in 1024x768 resolution, were investigated on an emotionally neutral Greek text similar to one used by Hill & Scharff (Hill & Scharff, 1997). These are:

- Plain text in font sizes 9pt, 10pt, 11pt, 12pt, 13pt, 14pt, 15pt, 16pt, 18pt, 26pt and 32pt and font type Times New Roman.
- Plain, bold, italics, bold-italics in size 16pt. in both Arial and Times New Roman, in color combinations as proposed in (Laarni, 2003) and (Hill & Scharff, 1997) (see Table 1).

During the experimental procedure, the presentation of some stimuli was repeated for two or three times, in order to correlate the dependency of the previously presented combinations of typographic elements to the current reader's emotional response. All stimuli were displayed in a random sequence for approximately 15 seconds. In the following screen participants were asked to assess their emotional state on a 9-point PAD scale using the manikins provided by the SAM test.

The duration of the experimental procedure was 20-30 minutes, depending on the participants response time.

5.3 Results

The results indicated that during the experimental procedure, the participants were mostly influenced by the font/background color combinations and the font size rather than by the typesetting elements and font type. For the later, a new experimental procedure should be designed and implemented that will clearly distinguish the affect of these elements to the emotional states.

Previously studied (Laarni, 2003) font and background color combinations as stimuli in the experimental procedure, was to investigate the hypothesis and the confirmation of the results that the SAM test is cross-cultural and language independent (Morris, 1995) test.

The mean values of the emotional states are displayed in Figure 6. According to (Laarni, 2003) and the presented experiment, the outcome can be summarized as:

- Red on Green (RG) color combinations are the most arousing and less pleasant.
- Black on White (BW) color combination has the lowest mean arousal value.
- White on Blue (WU), Green on Yellow (GY) and Black on White (BW) are the most pleasant combinations.

Abbreviation	Font Color	Background Color
YU	Yellow	Blue
WB	White	Black
GY	Green	Yellow
BW	Black	White
BG	Black	Grey
RG	Red	Green
WU	White	Blue

Table 1. The explaination of the abbreviations used in Fig. 6

The font size has a great impact to the reader's emotional state and especially to "Pleasure" and "Dominance" dimensions as presented in Figure 7. The "small-size fonts" (9 to 13 points) can be characterized as unpleasant and aroused. The "medium-size fonts" (14 to 27 points) are considered to be the most pleasant and calm. The "large-size fonts" (27 points and higher) have similar impact with "small size fonts".

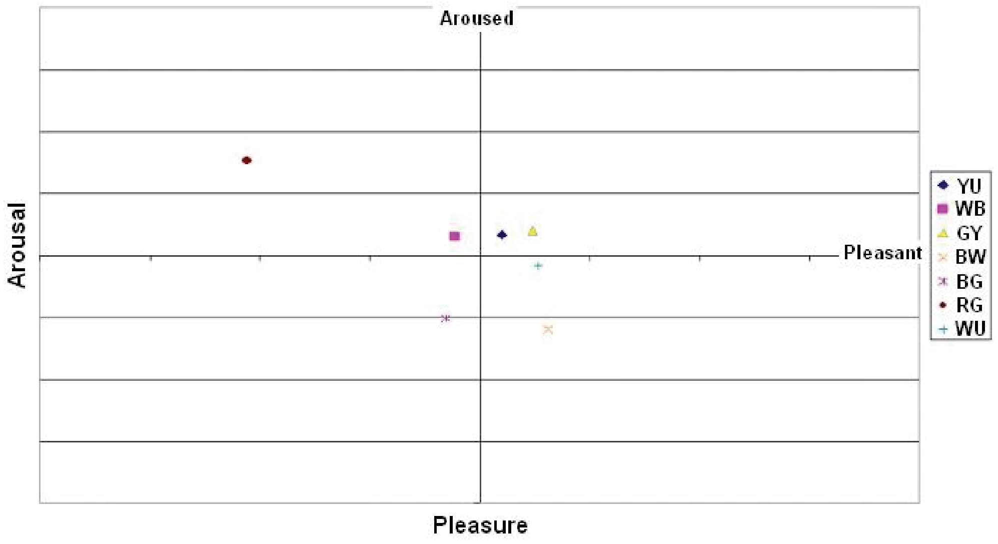


Fig. 6. The font and background color combinations on "Pleasure" and "Arousal" grid. The abbreviation of the color combinations are explained in Table 1

The statistical analysis of our results, is based on the approach used by (Grimm & Kroschel, 2005) on listeners' emotional state assessment. From equations (1), (2), (3) and (4), the resulting mean values of the participants-evaluators confidence scores ($\overline{r}_k$) are shown in Table 2.

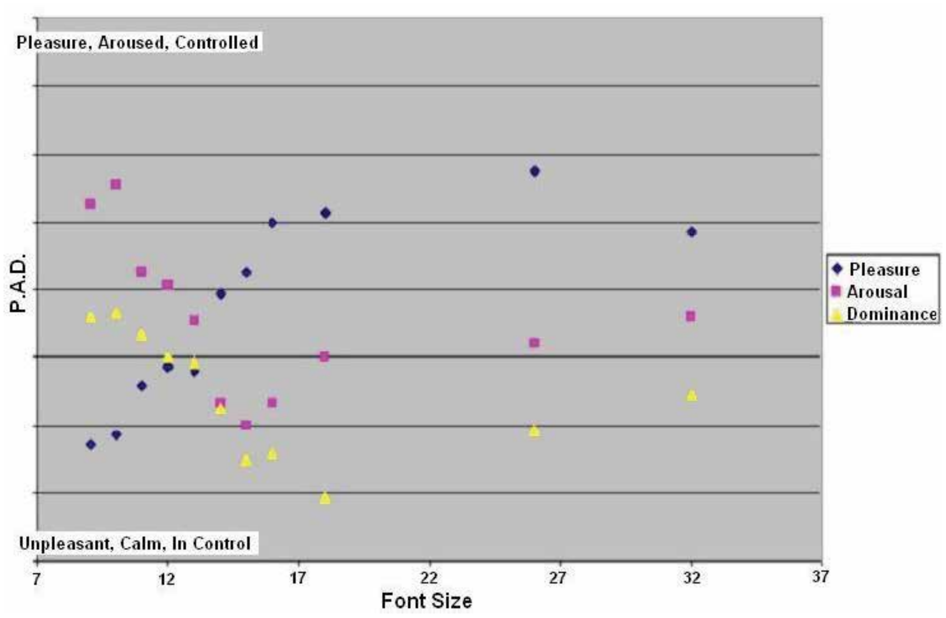


Fig. 7. The distribution of the impact of font size on the 3 dimensions of "Pleasure", "Arousal" and "Dominance"

	Mean Values			
Number of evaluators	Age	$\overline{r}_k^P$ for "Pleasure"	$\overline{r}_k^A$ for "Arousal"	$\overline{r}_k^D$ for "Dominance"
15	26,3	0,59	0,51	0,53

Table 2. The participants mean confidence score for each dimension

$$x_n^{MLE,(i)} = \frac{1}{K}\sum_{k=1}^{K} x_{n,k}^{(i)} \tag{1}$$

$$\mu_k^{(i)} = \frac{1}{N}\sum_{n=1}^{N} x_{n,k}^{(i)} \tag{2}$$

$$\mu^{MLE,(i)} = \frac{1}{N}\sum_{n=1}^{N} x_n^{MLE,(i)} \tag{3}$$

$$r_k = \frac{\sum_{n=1}^{N}(x_{n,k}^{(i)} - \mu_k^{(i)})(x_n^{MLE,(i)} - \mu^{MLE,(i)})}{\sqrt{\sum_{n=1}^{N}(x_{n,k}^{(i)} - \mu_k^{(i)})^2}\sqrt{\sum_{n=1}^{N}(x_n^{MLE,(i)} - \mu^{MLE,(i)})^2}} \tag{4}$$

An ideal evaluator is considered with $r_k^{(i)} = 1$ and an unreliable one with $r_k^{(i)} < 0$. All evaluators are reliable ($r_k^{(i)} > 0$).

6. Conclusion, future work and potential applications

In this chapter we presented a theoretical survey on emotions and a methodology on how reader's emotional state can be modelled. The selection of the Self Assessment Manikin Test and the dimensional approach for emotions led to a procedure for modelling the reader's emotional states on typesetting and font elements of e-documents. Documents contain many more elements and their combinations as presented in 4.1 prior to the elements in our experiment, for further experimentation.

The proposed modelling can be used in Artificial Intelligence for a number of applications. For example, as proposed in (Tsonos et al., 2007b), we can use the emotionally annotated documents for the multimodal accessibility of documents, focusing on the acoustic modality. Expressive Speech Synthesis (IEEE, 2006) creates more physical results. The dimensional approach of the emotions is also used in speech synthesis (Tsonos et al., 2007b) (Schröder, 2006). In Figure 8 a proposed real-time system that automatically produces emotional annotation to documents and conveys the visual elements into acoustic modality using expressive speech synthesis is presented.

Future work includes the use of an e-TSA composer (Xydas & Kouroupetroglou, 2001a, 2001b) (Xydas et al., 2005) on the DEMOSTHeNES Text-to-Speech platform (Xydas & Kouroupetroglou, 2001c) and models for expressive speech synthesis as proposed by Schröder (Schröder, 2006), for acoustic rendition of emotionally annotated documents.

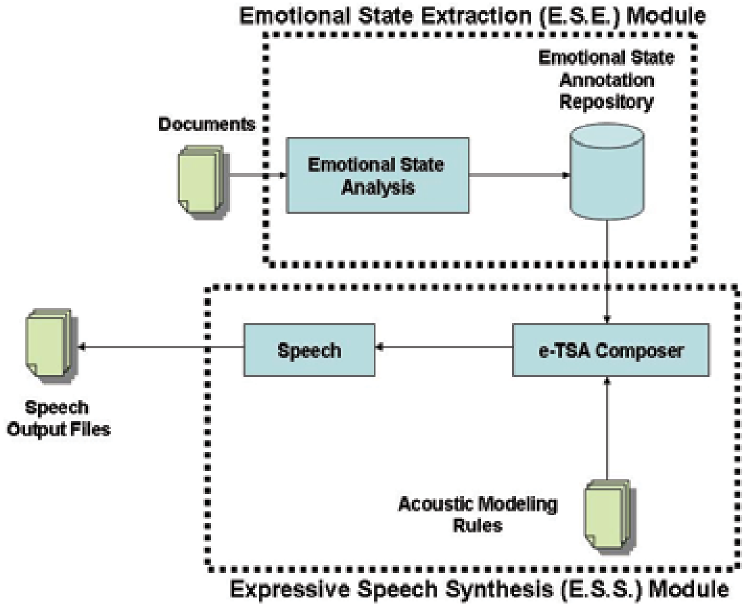

Emotional State Extraction (E.S.E.) Module
Emotional State Annotation Repository
Documents
Emotional State Analysis
Speech
e-TSA Composer
Speech Output Files
Acoustic Modeling Rules
Expressive Speech Synthesis (E.S.S.) Module

Fig. 8. The proposed architecture produces automatically emotional state annotation and uses Expressive Speech Synthesis for the acoustic production of documents

7. Acknowledgements

The work described in this chapter has been funded by the European Social Fund and Hellenic National Resources under the HOMER project of the Programme: PENED, Greek General Secretariat of Research and Technology.

8. References

ANSI/NISO (2008), Specifications for the digital talking book, *http://www.niso.org/kst /reports/standards/*

Bagozzi, R. P.; Gopinath, M. & Nyer, P. U. (1999). The Role of Emotions in Marketing. *Journal of the Academy of Marketing Science*, Vol. 27, No. 2, pp. 184 – 206, ISSN 1552-7824

Bänziger, T.; Tran, V., & Scherer, K. (2005). The Geneva Emotion Wheel: a tool for the verbal report of emotional reactions, *Poster presented at ISRE 2005*, Bari, Italy, (Available as pdf from http://www.unige.ch/fapse/emotion/isre/ BanzigerTranSchererISRE05.pdf)

Birren, F. (1984). *Color & Human Response: Aspects of Light and Color Bearing on the Reactions of Living Things and the Welfare of Human Beings*. J. Wiley & Sons Inc, ISBN: 978-0-471-28864-0, New York

Borchers, J.; Deussen, O.; Klingert, A. & Knörzer, C. (1996). Layout rules for graphical Web documents. *Computers and Graphics*, Vol. 20, No. 3, pp. 415 – 426, ISSN: 0097-8493

Busso, C.; Deng, Z.; Grimm, M.; Neumann, U. & Narayanan, S. (2007). Rigid Head Motion in Expressive Speech Animation: Analysis and Synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 15, No. 3, March 2007, pp. 1075-1086, ISSN: 1558-7916

Chul, M. L. & Narayanan, S. S. (2005). Toward detecting emotions in spoken dialogs. *IEEE Transactions on Speech and Audio Processing*, Vol. 13, No. 2, March 2005, pp. 293-303, ISSN: 1063-6676

Cornelius, R. R. (2000). Theoretical approaches to emotion, *Proceedings of ISCA Workshop on Speech and Emotion*, pp. 3–10, Belfast, Northen Ireland, September 2000

Conway, A. (1993). Page grammars and page parsing: a syntactic approach to document layout recognition, *Proceedings of the Second International Conference on Document Analysis and Recognition* 1993, pp. 761-764, ISBN: 0-8186-4960-7, Tsukuba Science City, Japan, 20 – 22 October 1993

DAISY Consortium (2008), *www.daisy.org*

Derrien-Peden, D. (1991). Frame-based system for macro-typographical structure analysis in scientific papers. *Proceedings of International Conference on Document Analysis and Recognition*, pp. 311-319, Saint-Malo, France, September 1991

Grimm, M. & Kroschel, K. (2005). Evaluation of natural emotions using self assessment manikins, *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 381-385, ISSN: 0-7803-9478-x, 27 Nov. – 1 Dec. 2005

Grimm, M.; Mower, E.; Narayanan S. & Kroschel, K. (2006). Combining categorical and primitives-based emotion recognition, *Proceedings of 14th European Signal Processing Conference (EUSIPCO)*, Florence, Italy, September 2006

Grimm, M.; Kroschel, K.; Mower, E. & Narayanan S. (2007a). Primitives-based evaluation and estimation of emotions in speech. *Speech Communication*, Vol. 49, No. 10-11, October 2007, pp. 787-800, ISSN: 0167-6393

Grimm, M.; Kroschel, K. & Narayanan, S.(2007b). Support Vector Regression for Automatic Recognition of Spontaneous Emotions in Speech, *IEEE International Conference on Acoustics, Speech and Signal Processing 2007 (ICASSP 2007)*, Vol. 4, pp. 1085-1088, ISBN: 1-4244-0728-1, Hawaii, USA, 15 – 20 April 2007

Hill, A. & Scharff, L. V. (1997). Readability of screen displays with various foreground/background color combinations, font styles, and font types, *Proceedings of the Eleventh National Conference on Undergraduate Research (NCUR-97)*, Vol. 2, pp. 742-746, Austin, Texas U.S.A., 24-26 April 1997

Holmberg, N. (2004). *Eye movement patterns and newspaper design factors. An experimental approach.* Master Thesis, Lund University Cognitive Science

Holmqvist, K.; Holsanova, J.; Barthelson, M. & Lundqvist, D. (2003). Reading or scanning? A study of newspaper and net paper reading, In: *The mind's eye: cognitive and applied aspects of eye movement research*, Hyönä, J. R., and Deubel, H. (Eds.), pp. 657-670, Elsevier Science Ltd, ISBN: 0-444-51020-6, Holland

Holmqvist, K. & Wartenberg, C. (2005). The role of local design factors for newspaper reading behaviour – an eye-tracking repspective, *Lund University Cognitive Studies*, 127. Lund: LUCS, ISSN: 1101-8453

IEEE (2006). Special Section on Expressive Speech Synthesis, *as presented in the Editorial of IEEE Transactions on Audio, Speech and Language Processing*, Vol. 14, No 4, July 2006, pp. 1097-1098, ISSN: 1558-7916

James, W. (1884). What is an emotion? *Mind*, Vol. 19, pp. 188 – 205, ISSN: 0026-4423

Fourli-Kartsouni, F. ; Slavakis, K. ; Kouroupetroglou, G. & Theodoridis S. (2007). A Bayesian Network Approach to Semantic Labelling of Text Formatting in XML Corpora of Documents, *Lecture Notes in Computer Science (LNCS)* Vol. 4556, pp. 299-308, ISBN : 978-3-540-73282-2

Krishnamoorthy, M.; Nagy, G.; Seth, S. & Viswanathan, M. (1993). Syntactic segmentation and labeling of digitized pages from technical journals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, Iss. 7, July 1993, pp. 737-747, ISSN: 0162-8828

Küpper, N. (1989). Recording of Visual Reading Activity: Research into Newspaper Reading Behaviour, (Available as pdf from *http://calendardesign.de/leseforschung/ eyetrackstudy.pdf*)

Laarni, J. (2003). Effects of color, font type and font style on user preferences, *Adjunct Proceedings of HCI International 2003*, C. Stephanidis (Ed.), Greece, pp. 31-32, Heraklion, University Press, ISBN: 960-524-166-8

Lang, P. J. (1985). The cognitive psychophysiology of emotion: Fear and anxiety, In: *Anxiety and the anxiety disorder*, A. Tuma & J. Maser (Eds.), pp. 131-170, Lawrence Erlbaum, New Jersey

Lang, P. J.; Bradley, M. M. & Culthbert, B. N. (2005). *International Affective Picture System (IAPS): Instruction Manual and Affective Ratings*. Technical Report A-6, The Center for Research in Psychophysiology, University of Florida, U.S.A.

Larson, K. (2007). The Technology of Text. *IEEE Spectrum*, May 2007, pp. 20 – 25, ISSN: 0018-9235

Mao, S; Rosenfeld, A. & Kanungo, T. (2003). Document structure analysis algorithms: a literature survey, *Proceedings of SPIE*, Vol. 5010, pp. 197-207, ISBN 0-8194-4810-9

Morris, J. D. (1995). Observations SAM: The self-assessment manikin- An efficient cross-cultural measurement of emotional response, *Journal of Advertising Research*, Vol. 6, November - December 1995, pp. 63-68, ISSN: 1470-7853

Narayanan, S. & Alwan, A. (2004). *Text to Speech Synthesis : new paradigms and advances*, Prentice Hall, ISBN: 9-780131-456617, Upper Sandle River, New Jersey

OASIS (2008). Organization for the Advancement of Structured Information Standards, *www.oasis-open.org/home/index.php*

Porat, T.; Liss, R. & Tractinsky, N. (2007). E-Stores Design: The Influence of E-Store Design and Product Type on Consumers' Emotions and Attitudes. *Lecture Notes in Computer Science (LNCS)*, Vol. 4553, pp. 712 - 721, ISBN: 978-3-540-73109-2

Richard, H. & Patrick, H. (2004). The Impact of Web Page Text-Background Colour Combinations on Readability, Retention, Aesthetics and Behavioural Intention. *Behaviour and Information Technology*, Vol. 23, No. 3, May-June 2004, pp.183-195, Taylor & Francis Group Journals, ISSN-0144-929X

Saari, T.; Turpeinen, M.; Laarni, J.; Ravaja N. & Kallinen, K. (2004). Emotionally Loaded Mobile Multimedia Messaging, *Third International Conference Entertainment Computing - ICEC 2004*, pp. 476-486, Eindhoven, The Netherlands, 1-3 September 2004

Sánchez, J. A.; Kirschning, I.; Palacio, J. C. & Ostróvskaya, Y. (2005). Towards mood-oriented interfaces for synchronous interaction, *Congreso Latinoamericano de Interacción Humano-Computadora (CLIHC´05)*, pp. 1-7, Cuernavaca, México, 2005

Sánchez, J. A.; Hernández, N. P.; Penagos J. C. & Ostróvskaya, Y. (2006). Conveying mood and emotion in instant messaging by using a two-dimensional model for affective states, *Proceedings of the Symposium on Human Factors in Computer Systems (IHC 2006)*, pp. 66-72, ISBN: 1-59593-432-4, Brazil, 2006, ACM, New York

Scherer, K. R. (2005). What are emotions? And how can they be measured? *Social Science Information*, Vol. 44, No. 4, pp. 693-727, ISSN: 0539-0184

Schröder, M. (2006). Expressing degree of activation in synthetic speech. *IEEE Transactions on Audio, Speech and Language Processing*, Vol. 14, Iss. 4, July 2006, pp. 1128-1136, ISSN: 1558-7916

Tsonos, D.; Xydas, G. & Kouroupetroglou G. (2007a). Auditory Accessibility of Metadata in Books: A Design for All Approach. *Lecture Notes in Computer Science (LNCS)*, Vol. 4556, pp. 436 - 445, ISBN: 978-3-540-73282-2

Tsonos, D.; Xydas, G. & Kouroupetroglou, G. (2007b). A Methodology for Reader's Emotional State Extraction to Augment Expressions in Speech Synthesis, *Proceedings of 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, Vol. 2, pp. 218-225, Patras, Greece, 29-31 October 2007 ISBN: 0-7695-3015-X

Tsonos, D.; Ikospentaki, K. & Kouroupetrglou, G. (2008). Towards Modeling of Readers' Emotional State Response for the Automated Annotation of Documents, *IEEE World Congress on Computational Intelligence (WCCI 2008)*, pp. 3252 - 3259, Hong Kong, 1 – 6 June 2008, ISSN: 978-1-4244-1821-3

Tsujimoto, S. & Asada, H. (1990). Understanding multi-articled document, *Proceedings of 10th International Conference on Pattern Recognition*, Vol. I, pp. 551-556, ISBN: 0-8186-2062-5, Atlantic City, NJ, USA, 16-21 June 1990

Wartenberg, C. & Holmqvist, K. (2005). Daily Newspaper Layout - Designers' Predictions of Readers' Visual Behaviour - A Case Study, *Lund University Cognitive Studies*, 126. Lund: LUCS, ISSN: 1101-8453

W3C (2008a), World Wide Web Consortium, *www.w3.org*

W3C (2008b), World Wide Web Consortium, Emotion Markup Language Incubator Group, *www.w3.org/2005/Incubator/emotion*

Xydas, G.; Argyropoulos, V.; Karakosta, T. & Kouroupetroglou, G. (2005). An Experimental Approach in Recognizing Synthesized Auditory Components in a Non-Visual Interaction with Documents, *Proceedings of the 11th Int. Conference on Human-Computer Interaction*, Las Vegas, Nevada USA, 22-27 July 2005, Lawrence Erlbaum Associates, Inc, ISBN: 0-8058-5807-5

Xydas G. & Kouroupetroglou, G. (2001a). Text-to-Speech Scripting Interface for Appropriate Vocalisation of e-Texts, *Proceedings of EUROSPEECH 2001*, pp. 2247-2250, Aalborg, Denmark, 2-7 September 2001, ISBN: 87-90834-09-7

Xydas, G. & Kouroupetroglou, G. (2001b). Augmented Auditory Representation of e-Texts for Text-to-Speech Systems, *Lecture Notes in Artificial Intelligence (LNAI)*, Vol. 2166, 2001, pp. 134-141, ISBN: 978-3-540-42557-1

Xydas, G. & Kouroupetroglou, G. (2001c). The DEMOSTHeNES Speech Composer, *Proceedings of the 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, pp. 167-172, Perthshire, Scotland, 29 August – 1 September 2001

Yamashita, A.; Amano, T.; Takahashi, I. & Toyokawa, K. (1991). A model based layout understanding method for the document recognition system, *Proceedings of International Conference on Document Analysis and Recognition*, pp. 130-138, Saint Malo, France, September 1991

Granule Based Inter-transaction Association Rule Mining

Wanzhong Yang, Yuefeng Li and Yue Xu
Queensland University of Technology
Australia

1. Introduction

Association rule mining extends data mining in a number of ways. Han et al. (Han & Kamber, 2006) summarized the features of association mining as single (Agraw et al., 1994) or multi-dimensional (Lee et al., 2006), multiple level (Han et al., 1999), quantitative (Ruckert et al., 2004), sequential pattern (Tzvetkov, P. et al., 2003), constraint based (Pei et al., 2001) etc. However, in line with requirements of knowledge discovered in the real applications, people also divide the association rule mining into intratransaction and intertransaction.
Intratransaction was the focus for most traditional approaches which include two stages of frequent itemset mining and rule generation. In frequent itemset mining, Apriori-like (Agraw et al., 1993) and FPT (Han et al., 2000) are two important methods applied in association rule mining. In a transaction database, intratransaction discusses the association of items in the same transaction. For example, if intratransaction association mining is applied in a market basket analysis, the high profit products could be associated with the low profit products in the same transaction.
Beyond intratransaction association mining, Lu et al. (2000) proposed intertransaction association mining which is the most widely used in some industry areas. Intertransaction association mining is used to discover patterns among different transactions. In the example of McDonalds, Burger King and KFC, the business association is along locations and also the time dimension as well.
Obviously, knowledge discovery in intertransaction association mining is more complicated than intratransaction association mining. To solve intertransaction association mining, Lu et al. (2000) proposed the EH-Apriori and E-Apriori algorithms. However, both algorithms take a significant amount of time to generate large extended itemsets.
Tung et al. (2003) proposed the FITI (First Intratransaction Then Intertransaction) algorithm which is adaptive to intertransaction association mining. It focuses on the two stages of mining frequent intertransaction itemsets and rule generation. The experimental results show this method is better for real world applications when compared to the high running time of EH-Apriori.
However, finding frequent intertransaction itemsets is still a time consuming process in intertransaction association mining. The FITI algorithm can show the efficiency when the transaction length is very short. If there are many items in each transaction and the threshold is low, the discovery process in frequent intertransaction itemsets could take a very long time. This algorithm considers the intervals of the intertransaction, but does not

include the average size of transactions. The extended database produces many unneeded combinations of items. The number of extended itemsets is much larger than the number of the set of items. It is difficult to apply this technique in high dimensional data.
It is a challenging issue to find the association among high dimensional data in industry. To breakthrough the classical methods in mining association rules, Li et al. (2006) proposed multi-tier granule mining for intratransaction association rules. Granule mining is a new initiative that attempts to improve the quality of discovered knowledge in multidimensional databases. The basic idea of granule mining came from decision tables presented by Pawlak (2002), where the attributes are divided by users into two groups: condition attributes and decision attributes; and a granule is a predicate that describes the common feature of a group of objects (transactions). In intratransaction association mining, granule mining has been proved to reduce the meaningless association rules significantly.
In this paper, we propose the method of granule based intertransaction association rule mining. It takes advantage of granule mining's capacity to transfer intertransaction association into a multi-tier structure and simplify the process of the long pattern in intertransaction association.
The remainder of the paper is structured as follows. We begin by introducing classical intertransaction association rule mining in Section 2. In Section 3, we introduce the concept of granule mining. In Section 4, we present granule based intertransaction association rule mining and use the multi-tier structure to deal with the process. In Section 5, we propose the concept of *precision* to evaluate the effectiveness of association rules and introduce the experimental results. The related work is in Section 6 and last Section contains the conclusion.

2. Intertransaction association mining

2.1 Intertransaction association mining

Let $T = \{t_1, t_2, \ldots, t_n\}$ be a transaction database, and each transaction is a set of items. Tung et al. (2003) used the sliding window and extended-items to describe the intertransaction. Each sliding window W can be viewed as a continuous ω (a fixed interval called maxspan, or *sliding_window_length*) sub-windows such that each sub-window contains only one transaction. Let e_i be an item, its occurrences in different transactions in a sliding window can be extended from $e_i(0)$ to $e_i(\omega)$, where $0, \ldots, \omega$ are positions of transactions in the window. The transactions in a sliding window W can be merged into a mega transaction (or extended transaction) by putting all of W's extended items in a collection. Hence, an inter itemset refers to a set of extended-items, and an inter association rule can be represented as $X \rightarrow Y$, where X and Y are both a set of extended-items and $X \cap Y = \varnothing$.
The definition of the support and confidence in inter association mining follows up the intra association mining. Let N be the number of megatransactions and, X and Y both be a set of extended-items and $X \cap Y = \varnothing$. Let T_{xy} be the set of megatransactions that contains X and also Y, and T_x be the set of megatransactions that contains X. We have

$$sup(X \rightarrow Y) = |T_{xy}|/N, \quad conf(X \rightarrow Y) = |T_{xy}| \,/\, |T_x| \, .$$

2.2 FITI algorithm

FITI algorithm is the state of the art in intertransaction association rule mining, it consists of three stages. For example 1 (in Table 1), we choose the same example used by Tung et al. But we assign the real meaning for each variable in ASX share market for bank and insurance. In

this example, there are only four transactions. Let *a, b, c* represent bank shares and *d, e, f, g, h, i* be insurance shares.
Stage I of FITI algorithm offers a data structure FILT to store frequent intra transaction items. Using FILT, stage II of FITI represents transaction database by FIT tables. Stage I and stage II is data preparation where FITI represents the database by an embedded data structure. We also can view the data preparation as the generation of an extended database. Stage III of FITI is data processing which generates frequent intertransaction itemsets. In this stage FITI controls the implementation using the algorithms.

d	*E*
100	a, b, c
104	a, b, c, d, e
105	a, f, g
109	e, h

Table 1. A database with four transactions

In stage I, the data structure, called Frequent-itemsets Linked Table (FILT) is used to store frequent intratransaction itemsets. This stage is based on the property that a frequent intertransaction itemset must be a frequent intratransaction itemset. To generate frequent intratransaction itemset, the Apriori algorithm is applied for the item selection. The basic structure of FILT for Table 1 is lookup links, where the frequent itemsets are sorted from one-itemset to n-itemset and stored in Table 2. This table can be extended to links in different directions. In table 2 *a, b, c* not only can be generator of {*a, b*}, {*a,c*} and {*b, c*}, but also can be the subset of {*a, b*}, {*a, c*} and {*b, c*}.

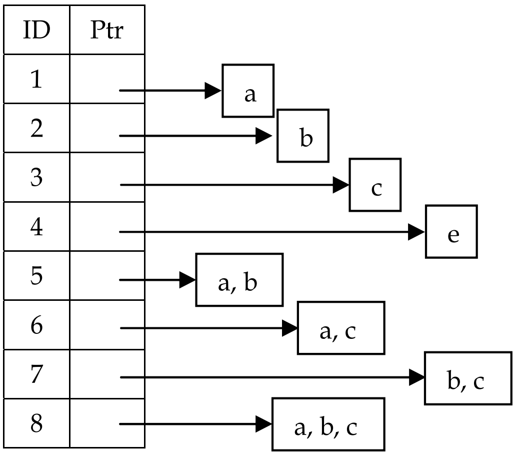


Table 2. Lookup links

Stage II of FITI is database transformation, where the database is represented by a set of *encoded Frequent-Itemset Tables* (**FIT** tables). A feature of FIT tables is that they represent each transaction by the ID of frequent itemsets in FILT. In particular, the FIT tables (in Table 3) can represent each transaction by the IDs from one-itemsets to *n*-itemsets separately.
Phase I and Phase II can be viewed as data preparation for frequent intertransaction itemsets mining. This process is mining frequent intratransaction itemsets and represents them by using encoded IDs sorted from one itemsets to *n*-itemsets.

F_1		*F_2*		*F_3*	
di	*$IDset_i$*	*di*	*$IDset_i$*	*di*	*$IDset_i$*
100	1, 2, 3	100		100	
104	1, 2, 3, 4	104	5, 6, 7	104	8
105	1	105	5, 6, 7	105	8
109	4	109		109	

Table 3. FIT tables

Stage III of FITI is data processing where FITI uses the algorithm to control intertransaction association and output a set of frequent intertransaction itemsets. The algorithm is based on the Apriori principle.

The input layer of the algorithm is the ID encoded intertransaction itemset *I*, which is corresponding to each sliding window. For the itemsets in each sub window, only one top ID from the FIT tables can be selected into *I* if it is available. Otherwise, it is zero.

During the data processing, the algorithm divides the generation of frequent inter-transaction itemset *I* into two cases of $k = 2$ and $k > 2$ in order to control the implementation.

When $k= 2$, FITI generates frequent intertransaction 2-itemsets, L_2. FITI makes use of the hashing approach in previous research and refines the hash formula for the bucket number.

When $k >2$, FITI uses the loop to generate the candidates of frequent intertransaction itemsets.

```
While ($L_{k-1} \neq \varnothing$ )
{
        Generate candidate intertransaction itemsets, $C_k$;
        Scan transformed database to update the count for $C_k$;
        Let $L_k = \{c \in C_k \mid support(c) \geq mins\ up\}$;
        $k$++;
}
```

The algorithm inside the loop is based on the classic Apriori principle which generates the itemsets based on the joins. However FITI separates the joins into intertransaction join and cross transaction join.

d	*E*
100	a
101	e, h
102	a, b
103	b, d
104	a
105	f, i
106	a, c
107	b, g

Table 4. A part of transaction database

W1	a(0) , b(1) f(1), a(2) d(2), g(3)
W2	a(0) h(0), b(1), i(2), e(3)

Table 5. Sliding windows

I	1, 0, 5, 2
J	1, 0, 6, 2

Table 6. Intertransaction joins

We use example 2 (in Table 4) to describe the intertransaction join in FITI. Table 4 is a part of the transaction database where the FIT tables are the same as Table 3. The sliding window size is four. So there are two sliding windows *W*1 and *W*2 for Table 4. The intertransaction itemsets in *W*1 and *W*2 are listed in Table 5. Their encoded IDs in the itemset of *I* and *J* are mapped from FIT tables listed in Table 6. According to the rule of intertransaction join, because 5 and 6 are in the same column of *I* and *J* in Table 6, and other items in *I* and *J* are same, the inter- transaction itemset is {1, 0, 8, 2}.

d	*E*
100	a
101	b f
102	a, d
103	g
104	a, h
105	b,
106	i
107	e

Table 7. A part of transaction database

W1	a(0) , e(1) h(1), a(2) b(2), b(3)d(3)
W2	a(0), f(1) i(1), a(2) c(2), b(3) g(3)

Table 8. Sliding windows

I	1, 2, 1, 0
J	1, 2, 0, 4

Table 9. Cross transaction join

We use example 3 (in Table 7) to describe cross transaction join in FITI. Table 7 is a part of the transaction database where the FIT tables are the same as Table 3. The sliding window size is four. So there are two sliding windows *W*1 and *W*2. The cross transaction itemsets in W1 and W2 are listed in Table 8. Their encoded IDs in the itemset of *I* and *J* are mapped from FIT tables listed in Table 9. According to the rules of cross transaction join, because {1, 0} and {0, 4} are in the last two columns of *I* and *J* in Table 8, and the values in the other columns are same. The intertransaction itemset is {1, 2, 1, 4}.
The next step in the algorithm is to calculate the count of each itemset and sort the itemset by the frequency. Following the Apriori algorithm, the process prunes off the itemsets which are not frequent and generates all frequent itemsets.

2.3 The restriction of FITI algorithm

The advantage of the FITI algorithm is is that it provides a complete set of frequent itemsets. However, the weakness is obvious as well. The FITI algorithm generates many redundant

itemsets. In particular in industry, those itemsets are useless. We use an example from the share market to illustrate this. Also, the large amount of calculation is time consuming.
FITI is based on the Apriori algorithm, which is used twice in mining inter- transaction association rules, including both generation of frequent intratransaction itemsets and generation of frequent intertransaction itemsets.
The advantage of Apriori is the output of a complete set of itemsets. The weakness is the complexity of the calculation. Therefore, intertransaction association is very complex. In particular when applying in high dimensional database, Apriori not only means a large amount of calculation for item joins, but also loss of the common feature in industry.
In the example in Table 1, each transaction has different length, where the number of items could be various. But in the share market, most important shares all have values on each transaction. We view all transactions as same length.
The FITI algorithm consists of three phases. The evaluation of the efficiency and effectiveness includes the three stages. We use the same example of bank and insurance in ASX. The example in Table 10 is a transaction database in ASX. There are over 250 working days in ASX each year and we view each working day as one transaction. Let *a*, *b*, *c* be the major banking shares and *d*, *e*, *f*, *g*, *h*, *i* represent the major insurance shares.

ID	a b c	d e f g h i
100	1 0 1	1 0 0 0 1 1
101	-1 0 1	1 0 1 -1 0 -1
102	0 -1 0	-1 0 1 0 0 -1
103	1 1 0	0 1 -1 0 -1 0

Table 10. Transaction database

In phase 1, FITI stores the intratransaction itemsets in the data structure FILT. In the ASX share market, each share has three statuses, booming, steady, dropping, where let 1, 0, -1 represent them separately. Let *a*1, *a*2, *a*3 represent the three status for *a*. With the same idea, we convert the transaction database into Table 11.

ID	a b c	d e f g h i
100	a1 b2 c1	d1 e2 f2 g2 h1 i1
101	a3 b2 c1	d1 e2 f2 g3 h2 i3
102	a2 b3 c2	d3 e2 f1 g2 h2 i3
103	a1 b1 c2	d2 e1 f3 g2 h3 i2

Table 11. Transformed database.

In FILT, if the transactions database is for one year, the max number of itemsets could be $3^9 = 19683$, where the length of FILT is hundreds times of the length ofASX transaction table. If let *N* be the length of transaction table, the length of FILT could be $O\ (N^2)$. Apparently many of itemsets in FILT are redundant for rule generation, which means a large amount of calculations are ineffective. This data structure expands the complexity of the algorithm and causes an expensive cost in industry.
The phase II of FITI is database transformation, which transforms the database into a set of FIT tables. FIT tables consist of one-item sets, two-item sets …. *N*-item sets FIT tables. Let *N* be the number of the FIT tables. If the transaction table is for one year, FIT tables could be from F_1 to F_{27}. If the status of each share becomes multiple choices in industry, the number

of FIT tables could be N times of the width of transaction database. Therefore, the input of the algorithm is a very large table set.

Moreover, the complexity extends into Phase III. The generation of candidate frequent intertransaction itemsets is based on the regulation of transaction joins. As an additional cost, we have to design many regulations for various joins. However, in Apriori algorithm we need to prune off the non-frequent itemsets. It causes a large amount of time.

To generate a complete set of itemsets, the length of FILT in the phase I and the width of FIT tables in phase II of FITI algorithm are N times of length and width of the transaction table separately. Therefore, the data structure become $O(N^2)$ times of the transaction database. We have to generate a large amount of meaningless itemsets. If we apply FITI algorithm in the high dimensional data set, the complexity is over reasonable estimation and the cost are so expensive. Apparently, it is not reasonable for the application in industry.

3. Granule mining

3.1 Decision tables and granules

In the multidimensional database, Pawlak proposed the decision tables in rough set theory to represent the association rules from the hidden patterns (Pawlak, Z., 2002) (Pawlak, Z., 2003). A feature of decision tables is related to user constraints, which divide the attributes of a database into condition attributes and decision attributes, respectively. We call the tuple (T, V^T, C, D) a *decision table* of (T, V^T) if $C \cap D = \emptyset$ and $C \cup D \subseteq V^T$, T is a set of transactions, and V^T is the set of attributes (items). The condition attributes C represent the premise (antecedent) of association rules, while the decision attributes D can be interpreted as the post-condition (consequent) of association rules.

In a decision table, there is a function for every attribute $a \in V^T$ such that a: $T \to V_a$, where V_a is the set of all values of a. We call V_a the domain of a. C (or D) determines a binary relation $I(C)$ (or $I(D)$) on T such that $(t_1, t_2) \in I(C)$ if and only if $a(t1) = a(t2)$ for every $a \in C$, where $a(t)$ denotes the value of attribute a for object $t \in T$. It is easy to prove that $I(C)$ is an equivalence relation, and the family of all equivalence classes of $I(C)$, that is a partition determined by C, is denoted by T/C.

The classes in T/C (or T/D) are referred to *C-granules* (or *D-granules*). The class which contains t is called *C-granule* induced by t, and is denoted by $C(t)$.

Table 12 simulates a part of the daily transactions for product sales in a shop, which is a multidimensional database. There are 200 transactions for 7 different products in the

Granule	Department	Commodity	Profit (%)	N_g
1	F & V	Accessories	Over 70	47
2	Bakery	General Merch	30-40	12
3	Bakery	Glassware	20-30	48
4	Soft Drinks	Dinners Frozen	Over 70	12
5	F & V	Accessories	30-40	21
6	Bakery	General Merch	20-30	40
7	Soft Drinks	Dinners Frozen	20-30	20

Table 12. A decision table

database. The possible attributes are *department, commodity, cost, price, profit*. The users choose only three attributes and let C = {*department, commodity*} and D = {*profit*}. We compress the database into a decision table, where each product is viewed as a granule and N_g is the number of transactions that belong to the granule.
Using Table 12 we can classify the condition granules (*C-granules*) as T/C = {{1,5}, {2,6},{3}, {4,7}} and decision granules (*D-granule*) as T/D = {{1,4}, {2,5}, {3,6,7}}, respectively.

Time	Products
t_1: 02/12/2003	a_1, a_5, a_6, a_7
t_2: 02/01/2004	a_1, a_5, a_6, a_7
t_3: 02/02/2004	a_1, a_2, a_4, a_5, a_7
t_4: 02/03/2004	a_1, a_2, a_3, a_5, a_6
t_5: 02/04/2004	a_1, a_2, a_4, a_5, a_6
t_6: 02/05/2004	a_1, a_2, a_3, a_5, a_6
t_7: 02/06/2004	a_1, a_4, a_5, a_7

Table 13. A time slice transaction table

Granule	**a_1**	**a_2**	**a_3**	**a_4**	**a_5**
cg_1	1	0	0	0	1
cg_2	1	1	0	1	1
cg_3	1	1	1	0	1
cg_4	1	0	0	1	1

(a) *C-granules*

Granule	**a_6**	**a_7**
dg_1	1	1
dg_2	0	1
dg_3	1	0

(b) *D-granules*

Granule	**a_1**	**a_2**	**a_3**	**a_4**	**a_5**	**a_6**	**a_7**	**N_g**
g_1	1	0	0	0	1	1	1	2
g_2	1	1	0	1	1	0	1	1
g_3	1	1	0	1	1	1	0	1
g_4	1	1	1	0	1	1	0	0
g_5	1	0	0	1	1	0	1	2

(c) Decision Table

Table 14. Granules

We also can view the transactions in Table 13, where the transactions come from the different time slices during 7 months and all products are frequent. Let $V^T = \{a_1, a_2, \ldots, a_7\}$, $T = \{t_1, t_2, \ldots, t_7\}$. According to the profit, *bananas Cavendish* (a_1), *coca cola 2LT* (a_2), *1.25 LT* (a_3), *chicken pieces* (a_4) and *potatoes brushed* (a_5) are all high profit products; *bread white* (a_6) and *sandwich* (a_7) are both low profit products. We set up the user constraint with the profit and classify the products into two groups. Let a_1, a_2, a_3, a_4, a_5 be condition attributes and a_6, a_7 decision attributes. Table 14 (a) is the *C-granules*; Table 14 (b) is the *D-granules*; Table 14 (c) is the decision table, where $T/C \cup D = \{g_1, g_2, g_3, g_4, g_5\}$ and N_g is the number of objects in the same granule.

In the representation of association rule mining, we view a decision table as a multidimensional database. Every granule in the decision table can be mapped into a decision rule (Pawlak, 2002). The condition attribute can be viewed as the premise of association rules; the decision attributes can be viewed as the post-conditions. The presence or absence of items is viewed as the same position. Therefore, we can obtain 5 decision rules in Table 14 (c), and the first one can be read as the following decision rule:

$$(a_1 = 1) \wedge (a_2 = 0) \wedge (a_3 = 0) \wedge (a_4 = 0) \wedge (a_5 = 1) \rightarrow (a_6 = 1) \wedge (a_7 = 1)$$

or in short $C(g_1) \rightarrow D(g_1)$ (or $C(t_1) \rightarrow D(t_1)$), where ^ means *"and"*.

From the above examples, we can now interpret association rules based on granules rather than patterns. In particular, we can view the association rules based on different granularities of multidimensional databases according to what users want.

3.2 Data mining and granule mining

Decision tables only provide a straightforward way to represent association rules. They only cover some kinds of larger patterns, but avoid many of the frequent patterns.

As the representation of association rule mining, we need to understand the difference between the patterns used in the decision tables and the association rules. To interpret this puzzle, we present the concept of decision patterns. Therefore, we need to define a series of concepts for illustrating the decision patterns.

Definition 1. A set of items X is referred to as an *itemset* if $X \subseteq V^T$. Let X be an itemset, where $[X]$ denotes the *covering set* of X, which includes all objects t such that $X \subseteq t$, i.e., $[X] = \{t \mid t \in T, X \subseteq t\}$.

Given an *itemset* X, its occurrence frequency is the number of objects that contain the *itemset*, that is $|[X]|$; and its support is $|[X]|/|T|$. An itemset X is called a *frequent pattern* if its support $\geq$ *min_sup* is a minimum support.

Definition 2. Given a set of objects Y, its itemset which satisfies

$$\text{itemset}(Y) = \{a \mid a \in V^T, t \in Y => a \in t\}.$$

Given a frequent pattern X, its *Closure*

$$Closure(X) = itemset([X]).$$

From the above definitions, we have the following theorem (Zaki, 2004).

Theorem 1. Let X and Y be frequent patterns. We have

$Closure(X) \supseteq X$ for all frequent patterns X;	(1)
$X \subseteq Y$ => $Closure(X) \subseteq Closure(Y)$.	(2)

Definition 3. A frequent pattern X is *closed* if and only if $X = Closure(X)$.

Given a *C-granule* $cg = C(t)$, its *covering set* $[cg] = \{t' \mid t' \in T, (t', t) \in I(C)\}$. Let cg be a *C-granule* and dg be a *D-granule*, we define $[cg \wedge dg] = [cg] \cap [dg]$.For example, in Table 14 $g_1 = \{(a_1 = 1) \wedge (a_2 = 0) \wedge (a_3 = 0) \wedge (a_4 = 0) \wedge (a_5 = 1) \wedge (a_6 = 1) \wedge (a_7 = 1)\} = C(g_1) \wedge D(g_1) = cg_1 \wedge dg_1$; therefore

$$[g_1] = [cg_1 \wedge dg_1] = [cg_1] \cap [dg_3] = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\} \cap \{t_1, t_2\} = \{t_1, t_2\}.$$

Table 15 illustrates the covering sets of granules, where (a) includes the covering sets of *C-granules*, (b) includes the covering sets of *D-granules*, and (c) includes the covering sets of *C∪D-granules*.

Theorem 2. Let (T, V^T, C, D) be a *decision table*. We have

$[C(t)] \supseteq [C \cup D(t)]$, for all $t \in T$.	(1)
The derived decision pattern of every granule $g \in T/C \cup D$ is a closed pattern.	(2)

Granule	a1	a2	a3	a4	a5	covering set
cg1	1	0	0	0	1	{t1, t2, t3, t4, t5, t6, t7 }
cg2	1	1	0	1	1	{ t3, t5}
cg3	1	1	1	0	1	{ t4, t6}
cg4	1	0	0	1	1	{ t7 }

(a) *C-granules*

Granule	a6	a7	covering set
dg1	1	1	{ t1, t2}
dg2	0	1	{t3 , t7 }
dg3	1	0	{t4 , t5 , t6 }

(b) *D-granules*

Granule	a1	a2	a3	a4	a5	a6	a7	Ng	Covering set
g1	1	0	0	0	1	1	1	2	{ t1 , t2 }
g2	1	1	0	1	1	0	1	1	{ t3 }
g3	1	1	0	1	1	1	0	1	{ t5 }
g4	1	1	1	0	1	1	0	0	{t4, t6 }
g5	1	0	0	1	1	0	1	2	{ t7 }

(c) *Decision Table*

Table 15. Covering set of C∪*D -granules*

Proof: (1) is obvious in accordance with the definition of closure.

For (2), Let X be the derived pattern of g, that is, $X=\{a_i \in C \cup D \mid a_i(g) = 1\}$. From the definition of the granules, we know there is an object $t_0 \in [g]$ such that $X = \{a_i \in C \cup D \mid a_i(t_0) = 1\}$, that is $t_0 \in [X]$.

Given an item $a \in itemset([X])$, according to Definition 2 we have $a \in t$ for all $t \in [X]$, that is, $a \in t_0$ and also $a \in X$. Therefore, $Closure(X) = itemset([X]) \subseteq X$.

We also have $X \subseteq Closure(X)$ from Theorem 1, and hence we have $X = Closure(X)$.

4. Granule based intertransaction association rule mining

Formally a transaction database can be described as an information table ($\mathcal{D}$, $V^{\mathcal{D}}$), where $\mathcal{D}$ is a set of objects in which each object is a sequences of items, and $V^{\mathcal{D}} = \{a_1, a_2, \ldots, a_n\}$ is a set of selected items (or called attributes) for all objects in $\mathcal{D}$.

Decision tables are efficient for dealing with multiple dimensional databases in line with user constraints. Formally, users may use some attributes of a database; and they can divide these attributes into two target groups: condition attributes and decision attributes, respectively. We call the tuple ($\mathcal{D}$, $V^{\mathcal{D}}$, C, D) a decision table of ($\mathcal{D}$, $V^{\mathcal{D}}$) if $C \cap D = \emptyset$ and $C \cup D \subseteq V^{\mathcal{D}}$. The classes in $\mathcal{D}/C$ (or $\mathcal{D}/D$) are referred to C-granules (or D-granule).

For example, in the share market, a transaction contains different shares at the same day. To reduce the risk of investments, share-market experts usually consider a group of shares rather one or two shares based on the current performance of another group of shares. To help such investments, we can group shares into different industry categories. For instance, we may choose two industries: bank and insurance.

The mining process has three sub stages.

1. Transform the transaction database into the form of a decision table;
2. Generate C-granules and D-granules based user selected two industry categories;
3. Generate inter association rules between C-granules and D-granules.

The original transaction database records the data of ASX share transactions along the date dimension. The data includes attributes like high, low, open and close, which represent the price status in a day. To keep up the monotonic property, we assume the transactions are continuous and all records are complete filled. The empty records are instead of null value.

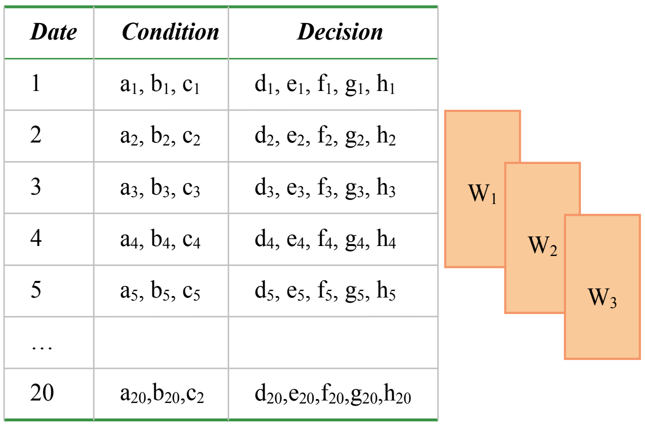
Since the mining object is transferred from the item to the group, a sliding window not only considers an interval (*sliding_window_length*), but also the number of attributes (we call *sliding_window_width*).

When transforming the transaction database to the decision table ($\mathcal{D}$, $V^{\mathcal{D}}$, C, D), let the banking shares be condition attributes C and the insurance shares be decision attributes D.

We can use the normal way for dealing with C-granules. We use the technique of sliding windows to generate D-granules, where *sliding_window_width* = $|D|$. Let $\mathcal{D}$ be all the transactions and V_a refers to the profit gain of all shares in each transaction. V_a includes three statuses: increased, neutral and loss, represented by 1, 0 and -1.

In Figure 1 there are three bank shares a, b, c as condition attributes that represent *Westpac* bank, *ANZ* bank and *National* bank separately. Let a_i, b_i, c_i be the profit gain of bank shares on day i. The decision attributes d, e, f, g, h represent insurance shares *PMN*, *IAG*, *AMP*, *QBE*, *AXA*, where d_i, e_i, f_i, g_i, h_i refer to the profit gain of insurance shares on day i. The sliding windows only contains decision attributes, and the *sliding_window_width* =5 and *sliding_window_length*=3. The interval of the transactions decides the block of transactions in the sliding window, which would be used to generate D-granules for a same C-granule.

To describe the inter associations between condition granules and decision granules, we can extend the normal decision table into an extended decision table such that each condition granule is linked to all possible sub-windows in sliding windows. For example, Table 16 illustrates an extended decision table when we let *sliding_window_length* = 2.



Date	*Condition*	*Decision*
1	a_1, b_1, c_1	d_1, e_1, f_1, g_1, h_1
2	a_2, b_2, c_2	d_2, e_2, f_2, g_2, h_2
3	a_3, b_3, c_3	d_3, e_3, f_3, g_3, h_3
4	a_4, b_4, c_4	d_4, e_4, f_4, g_4, h_4
5	a_5, b_5, c_5	d_5, e_5, f_5, g_5, h_5
…		
20	a_{20},b_{20},c_2	d_{20},e_{20},f_{20},g_{20},h_{20}

Figure 1. A decision table with sliding windows

ID	Condition	Decision
1	a_1,b_1,c_1	d_2,e_2,f_2,g_2,h_2
2	a_1,b_1,c_1	d_3,e_3,f_3,g_3,h_3
3	a_2,b_2,c_2	d_3,e_3,f_3,g_3,h_3
…		
39	a_{20},b_{20},c_{20}	d_{21},e_{21},f_{21},g_{21},h_{21}
40	a_{20},b_{20},c_{20}	d_{22},e_{22},f_{22},g_{22},h_{22}

Table 16. An extended decision table with maxspan = 2

The data compression is along the vertical direction in the extended decision table. Let $\mathcal{D}/C$ be the set of C-granules that refer to all classes of the profit situations for three bank shares. Let $\mathcal{D}/D$ be the set of D-granules that refer to all classes of the profit situations for five insurance shares. The inter association rule mining can be represented by mining granules now.

It is hard to clearly understand the intertransaction associations between condition granules and decision granules because of many duplicates. For this purpose we would like to represent the extended decision table as a 2-tier structure. The first tier contains all condition granules, the second tier contains decision granules and the intertransaction associations are the links.

For the above example, people concern the gain of the group of shares, not only single share. Therefore, we can use a simple *SUM* measure to denote the gain information of a group of shares, where $SUM > 0$ means positive gain, $SUM < 0$ means negative gain and $SUM = 0$ means no-gain.

Figure 2 depicts an example of a 2-tier structure, where we have seven condition granules that describe the possible changes of three bank shares; and have only three decision granules that describe the possible gains of buying five insurance shares after 1 or 2 days based on the changes of the three bank shares.

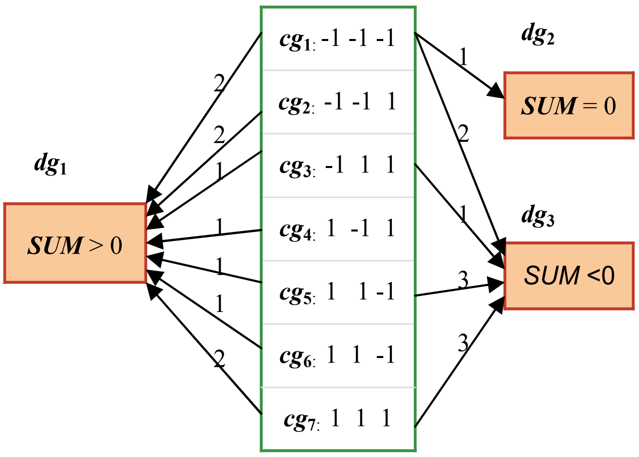


Figure 2. The association of *C* granules and *D* granules

Formally, a set of items X is referred to as an itemset if $X \subseteq V^{\mathcal{D}}$. Let X be a itemset, we use $[X]$ to denote the covering set of X, including all objects d such that $X \subseteq d$, i.e., $[X] = \{d \mid d \in D, X \subseteq d\}$.
Let $\mathcal{D}/C = \{cg1, cg\,2\,, \ldots, cg_m\}$ and $\mathcal{D}/D = \{dg1, dg2, dg3\}$. The decision rules in Figure 2 can be illustrated as follows:

$$cg_x \rightarrow dg_z$$

$$conf = |[cg_x \wedge dg_z]| / |cg_x|$$

$$support = |[cg_x \wedge dg_z]| / N$$

In Figure 2, there are twelve associations. If we set up the *min_sup* = 2, we have the following six inter association rules:

$$cg1 \rightarrow dg1\ (conf = 2/5) \quad cg1 \rightarrow dg3\ (conf = 2/5)$$

$$cg2 \rightarrow dg1\ (conf = 2/2) \quad cg5 \rightarrow dg3\ (conf = 3/4)$$

$$cg7 \rightarrow dg1\ (conf = 2/5) \quad cg7 \rightarrow dg3\ (conf = 3/5)$$

5. Experiments

5.1 Basic experiments

In the ASX share market, there are 26 industries and almost 2000 companies. We take the ASX data of four industries from January 2005 to January 2007. We divide the data into two sections: a training set and a testing set. The first section contains over 260,000 transactions in 2005. The second section includes over 340,000 transactions in the other.
We choose two pairs of industries for the experiments: bank vs. insurance and food beverage & tobacco vs. retailing. In each pair, according to the yearly share volumes, we select the top three shares of one industry as condition granules and the top five products of another industry as decision granules.

Table 17 describes some samples for the first pair of industries in 2005 and the interval is one day. There are 15 condition granules. In the second pair of industries, there are 23 condition granules. The constraint-based decision granules are decided base on $SUM > 0$, $SUM = 0$ and $SUM < 0$. We choose three intervals for inter association mining. The intervals are one day, two days and three days.

ID	B1	B2	B3	SUM>0	SUM=0	SUM<0
1	-1	-1	-1	27	6	27
2	-1	-1	0	5	0	0
3	-1	-1	1	12	1	11
…						
15	1	1	1	33	5	29

Table 17. Bank vs. insurance in 2005

5.2 Precision

When applying the inter association rule in the real data, we propose *Precision* as the criterion to evaluate the effectiveness of inter association rules.

In share market, investors should be interested in the prosperous shares where $SUM \geq 0$. Let $cg_x \rightarrow dg_z$ be an inter association rule discovered in training phase and $SUM(dg_z) > 0$, a positive gain.

Let S_{fst} be the number of transactions in the testing set that match cg_x. Let S'_{snd} be the number of dg_z with $SUM(dg_z) \geq 0$ that match cg_x , and S_{snd} be the number of dg_z with $SUM(dg_z) > 0$ that match cg_x.

We define P_N as *Non_ Negative_ Precision* where

$$P_N(cg_x \rightarrow dg_z) = (S'_{snd} \; / \; S_{fst}) * 100\% \; .$$

We also define P_P as *Positive_ Precision* where

$$P_P\,(cg_x \rightarrow dg_z) = (S_{snd} \; / \; S_{fst}) * 100\%.$$

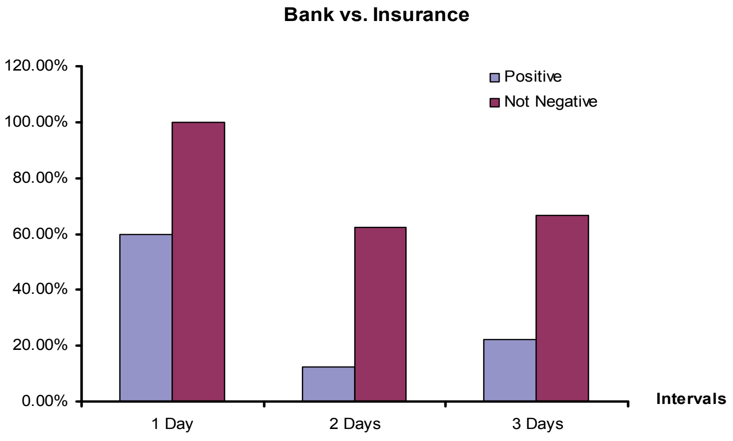


Figure 3. Precision for bank vs. insurance

In Figure 3 the pair is bank and insurance. All *Non_Negative_Precisions* are between 60% and 100%. All *Positive_Precisions* are greater than 10%. When the interval is one day, the positive percentage reaches 60%.

5.3 Efficiency

Compared to the FITI algorithm, granule-based inter association mining makes long pattern mining possible and easier. In the FITI algorithm, the max frequent patterns of eight items in Figure 1 listed in Figure 4 N_P = 28 = 256. It expands the scope of the user requirement and generates many extra items. In the basic experiments, each pair includes eight different frequent items. In both pairs of industries, the minimum numbers of association rules are 15 and 23 separately; the maximum numbers of association rules are 45 and 69 separately. Our method obviously reduces the time and looks more efficient and applicable in the above example.

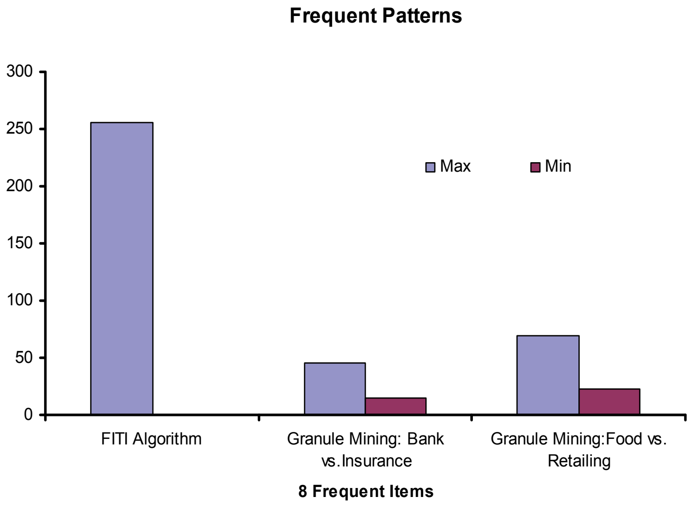


Figure 4. Frequent Patterns

In addition, the granule based approach has advantage in common feature instead of FITI based on item association for decision rule generation. It avoids of generation of the extended database and the joins in Apriori-like algorithms. As knowledge representation, granule based approach simplifies the complexity of the algorithms and keeps the major benefit of intertransaction association mining, which is easy to understand and use. In particular, it is more practical and meaningful in industry. However, the weakness is the loss of the completeness of item association.

For the future research, we need to develop granule based intertransaction association rule mining in two aspects. At first we will keep studying how to apply granule based intertransaction association rule in high dimensional data. Secondly we need to consider how to use multi-tier structure to improve the quality of rules during the rule discovery. Also the data scope of the experiments will expand to various share products running for long term. Moreover, the design idea can extend to different fields in industry.

6. Related work

Granule based intertransaction association rule mining is involved in intratransaction association rules and granule mining. The related work is for both issues.
Most current researchers endeavor to use the existing efficient algorithms for mining intratransaction association rules. Apriori like algorithm (Agraw et al., 1993) and FP-tree algorithm (Han et al., 2000) are two foundation methods in this field. To apply association rule mining in industry, the association mining scope expands from single dimensional association to multidimensional association, even extending to the multilevel.
However, to satisfy complex requirements in industry, intertransaction association rule mining looks attractive. Lu et al. (2000) first presented the concept of intertransaction association rule mining and contributed E-Apriori and EH-Apriori algorithms. The performance of these algorithms suffered when dealing with real data in industry. To speed up the above process, Feng et al. (2002) presented a template model that includes several optimization techniques, i.e., joining, converging.
Moreover, Tung et al. (2003) proposed the FITI algorithm to overcome the shortcoming in previous methods. Also FITI turns to be a brilliant milestone in intertransaction association rule mining. FITI algorithm offers the data structure FILT to store frequent intratransaction itemsets and transfer database into FIT tables as input for data processing, where FITI algorithm generates frequent intertransaction itemsets by joins. Mining both intra and inter frequent itemsets are all based on Aprior algorithm. The nature of FITI algorithm seems an extension of Apriori like idea in interaction association. Because of the complexity of the interaction association, the disadvantage of Apriori causes many extra itemsets during the joins. It is difficult to cope with the long patterns in the intertransaction.
To improve the efficiency of intertransaction association, few methods recently are proposed. In intertransaction frequent closed itemsets algorithm (IFCIA) (Dong et al. 2007), the basic design follows up FITI algorithm. The contribution is the closed itemsets, which are applied in mining process in order to avoid of the extended database. But it is still in the frame of Apriori like scope. MMIT is the interaction based on matrix mining (Zhang et al., 2007). In the algorithm design, MMIT is different from FIT in two aspects. First, MMIT directly moves into mining and sorting of intertransaction itemsets at the first step. It avoids of intratransaction itemsets mining and Apriori like idea. Secondly, MMIT uses matrix for mining frequent intertransaction itemsets. However, the experiments seem not enough to prove this method.
Granule mining originally is from rough set theory (Pawlak, 1982). The rough set theory can be used to describe the knowledge in information tables (Guan et al. 2003; Li et al. 2003). Further, rough sets based decision tables presented by Pawlak (2002) can be used to represent some sorts of association rules. Li and Zhong (2003) presented a structure to disconnect the condition granules and decision granules in order to improve the efficiency of generating association rules from decision tables.
To cope with a multiple dimensional transaction database with current algorithms, rough set theory becomes more abstractive in association rule mining. Pawlak (2002) proposed the decision table and divided the transaction table into the condition attribute and the decision attribute. Li et al. (2003) presented a new algorithm to modify Pawlak's method and improved the efficiency. Both algorithms can make use of the advantage of rough sets, which can find minimal sets of data and generate minimal sets of association rules (Pawlak, 1996). However, they are only suitable for a low dimensional system. It does not offer a

solution for the decomposition of a high dimensional database. We also need to reduce dimensionality for a large database.

Li et al. (2006) presented a multi-tier structure for granule mining to represent multidimensional intratransaction association rules. It breakthroughs traditional methods in association rule mining. One feature of this approach is focusing on the association among the granules. The multi-tier structure can improve the quality of association rule mining and reduce attributes for a large database. Also this method can be applied in data processing in data warehouse (Yang et al., 2008).

7. Conclusion

In this chapter, we present granule based inter association mining to reduce the complexity of intertransaction association rule mining. To compare with other methods, our method can reduce the width of sliding windows. It uses granules to replace extended item sets. Thus, we do not need to consider too many combinations of extended items. We also propose the concept of precision in order to evaluate the effectiveness of intertransaction association rule mining. The experiments show that the proposed method is promising.

8. References

Agraw, R., Imielinski, T.&Swami, A. (1993). Mining association rules between sets of items in large database, *Proceedings of ACM-SIGMOD*, pp. 207-216, Montreal, Canada, 1993

Agraw, R., Srikant, R. (1994). Fast Algorithms for Mining Association Rules in Large Databases, *Proceedings of the 20th International Conference on Very Large Data Bases*, pp. 487-499, 1994

Dong, J. & Han, M. (2007). IFCIA: An Efficient Algorithm for Mining Intertransaction Frequent Closed Itemsets, *Proceedings of fourth international conference on fuzzy systems and knowledge discovery*, pp. 678-682, Haikou, China, 2007

Feng, L., Yu, J. X., Lu, H.&Han, J. (2002). A template model for multidimensional inter-transactional association rules, *The International Journal on Very Large Data Bases*, 11(2), (2002) pp. 153 -175

Han, J. & Fu, Y. (1999). Mining multiple-level association rules in large databases, *IEEE Transaction on Knowledge and Data Engineering*, Vol. 11, No. 5, pp. 798-805, 1999

Han, J. & Kamber, M. (2006). *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers

Han, J., Pei, J. & Yin, Y. (2000). Mining frequent patterns without candidate generation, *Proceedings of the 2000 ACM SIGMOD international conference on Management of data table of contents*, pp. 1-12, Texas, United States, 2000

Lee, A. J. T., Lin, W. & Wang, C. (2006). Mining association rules with multi-dimensional constraints, *The Journal of Systems and Software*, pp. 79-92, 2006

Li, Y. & Zhong, N. (2003). Interpretations of association rules by granular computing, *Proceedings of 3rd IEEE International Conference on Data Mining*, pp. 593-596, USA, 2003

Li, Y., Yang, W. & Xu, Y. (2006). Multi-Tier Granule Mining for Representations of Multidimensional Association Rules, *Proceedings of 6th IEEE International Conference on Data Mining*, pp. 953-958, Hong Kong, 2006.

Lu, H., Han, J. & Feng, L. (2000). Beyond intratransaction association analysis: mining multidimensional intertransaction association rules, *ACM Transactions on Information Systems*, 18(4), (2000) pp.423 – 454

Pawlak, Z. (1982) Rough Sets, *International Journal of Computer and Information Science*, Vol.11, No.5, (1982), pp. 341-356

Pawlak, Z. (1996). Rough sets and data analysis, *Proceedings of IEEE AFSS*, pp. 1-6, Kenting, Taiwan, 1996

Pawlak, Z. (2002). In pursuit of patterns in data reasoning from data, the rough set way, *Proceedings of 3rd International Conference on Rough Sets and Current Trends in Computing*, pp. 1-12, USA, 2002

Pawlak, Z. (2003). Flow graphs and decision algorithms, *Proceedings of 9th International Conference on Rough Set, Fuzzy Sets, Data Mining and Granular Computing*, pp. 1-10, Chongqing, China, 2003

Pei, J., Han, J. & Lakshmanan, L.V.S. (2001). Mining frequent itemsets with convertible constraints, *Proceedings of 17th International Conference on Data Engineering*, pp. 433-442, Heidelberg, Germany, 2001

Ruckert, U., Richter, L. & Kramer, S. (2004). Quantitative association rules based on half-spaces: an optimization approach, *Proceedings of fourth IEEE International Conference on Data Mining*, pp. 507 – 510, Brighton, UK, 2004

Tung, A.K.H., Lu, H., Han, J. & Feng, L. (2003). Efficient mining of intertransaction association rules, *IEEE Transactions on Knowledge and Data Engineering*, Vol.15, No.1, (2003), pp.43–56

Tzvetkov, P., Yan, X. & Han, J. (2003). TSP: mining top-K closed sequential patterns, *Proceedings of 3rd IEEE International Conference on Data Mining*, pp. 347-354, Urbana, IL, USA, 2003

Yang, W., Li, Y., Wu, J. & Xu, Y., Granule Mining Oriented Data Warehousing Model for Representations of Multidimensional Association Rules, *International Journal of Intelligent Information and Database Systems*, Vol.2, No.1, (2008), pp. 125-145 2008.

Yang, W., Li, Y. & Xu, Y. (2007). Granule Based Intertransaction Association Rule Mining, *Proceedings of 19th IEEE International Conference on Tools with Artificial Intelligence,* Vol.1, pp. 337-340, Patras, Greece, 2007

Zhang, Z., Wang, H. & Huang, G. (2007). A New Algorithm Based on Matrix for Mining Inter-Transaction Association Rules, *International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 6717-6720, Shanghai, China, 2007

Countering Good Word Attacks on Statistical Spam Filters with Instance Differentiation and Multiple Instance Learning

Yan Zhou, Zach Jorgensen and Meador Inge
University of South Alabama
USA

1. Introduction

Although the impact of e-mail spam has been greatly reduced by existing spam filters, spam remains a great challenge to Internet Service Providers and the average user. Given that there are millions of e-mail users, profit-driven spammers have great incentives to spam. With as little as 0.001% response rate, a spammer could potentially profit $25,000 on a $50 product (Carpinter and Hunt, 2006). Over the years, spammers have grown in sophistication with cutting-edge technologies and have become more evasive. The best evidence of their growing effectiveness is a recent estimate of over US $10 billion worldwide spam-related cost in terms of wasted resources and lost productivity (Jennings, 2005). The spam problem has been generally accepted as a long lasting problem, unlikely to end anytime soon.
Spam filtering has become an extensively studied subject due to the severity of the spam problem. However, relatively little research has been done on countering adversarial attacks on existing spam filtering systems. In recent years, adversarial attacks have become an increasing challenge to the anti-spam community. Common adversarial attacks on spam filters are exploratory attacks. These attacks attempt to discover ways to disguise spam messages so that spam filters are unable to correctly detect them. However, the adversary who initiates this type of attacks has no intention to influence or alter the training process of spam filters that are the targets of attack. The good word attack (Lowd and Meek, 2005b) is one of the most popular exploratory attacks employed by spammers. This technique involves appending to spam messages sets of "good" words that are common to legitimate e-mails (ham) but rare in spam. Spam messages injected with good words are more likely to bypass spam filters. So far, relatively little research has been done on how spam filters can be trained to account for such attacks. This chapter presents an effective defence strategy, using instance differentiation and multiple instance (MI) learning, against spam disguised with good words.
Multiple instance (MI) learning (Dietterich et al., 1997) differs from single instance supervised learning in that an example is represented by a set, or bag, of instances rather than as just a single instance. The bag is assigned a class label (either positive or negative) based on the instances it contains; however, the instances within the bag are not necessarily labelled. Classic MI learning assumes that a bag is positive if at least one instance in the bag is positive and negative if all instances are negative. Therefore, the goal of multiple instance learning is to learn a classification function that accurately maps a given bag to a class.

Our spam filtering strategy adopts the classical MI assumption. Each e-mail is transformed into a bag of instances. An e-mail is classified as spam if at least one instance in the corresponding bag is spam, and as legitimate if all the instances in it are legitimate. The other integral component of our defence strategy is the model for instance differentiation, i.e., dividing an e-mail into a bag of multiple instances. The challenge of defining such a model includes finding a clean separation between the spam content and the good words that are common to legitimate e-mail, and ensuring that the model itself does not introduce a loophole for adversarial attacks. With an effective instance differentiation model, we can split each e-mail into multiple instances so that even when spam is injected with good words, a multiple instance learner is still able to recognize the spam part of the message.
In this chapter, an overview of recent research in the area of adversarial learning and multiple instance learning is provided. The models for performing instance differentiation and multiple instance learning are presented and investigated. Experimental results are given to show that a multiple instance learner stands up better to good word attacks than its single instance counterpart and the commonly practiced Bayesian filters. Then further development and future directions for this research are discussed.

2. Related work

Our work is primarily motivated by recent research on adversarial learning (Dalvi et al., 2004; Lowd & Meek, 2005a; Kolter & Maloof, 2005). Dalvi et al. (2004) consider classification to be a game between classifiers and adversaries in problem domains where adversarial attacks are expected. They model the computation of the adversary's optimal strategy as a constrained optimization problem and approximate its solution based on dynamic programming. Subsequently, an optimal classifier is produced against the optimal adversarial strategy. Their experimental results demonstrate that their game-theoretic approach outperforms traditional classifiers in the spam filtering domain. However, in their adversarial classification framework, they assume both the classifier and the adversary have perfect knowledge of each other, which is unrealistic in practice.
Instead of assuming the adversary has perfect knowledge of the classifier, Lowd & Meek (2005a) formalized the task of adversarial learning as the process of reverse engineering the classifier. In their adversarial classifier reverse engineer (ACRE) framework, the adversary aims to identify difficult spam instances (the ones that are hard to detect by the classifier) through membership queries. The goal is to find a set of negative instances with minimum adversarial cost within a polynomial number of membership queries. Newsome et al. (2006) emphasize the point that the training data used to build classifiers for spam filtering and the similar problem of internet worm detection is, to a large extent, controlled by an adversary. They describe and demonstrate several attacks on the generators of such classifiers in which the adversary is able to significantly impair the learning of accurate classifiers by manipulating the training data, even while still providing correct labels for the training instances. The attacks involve inserting features, in a specific manner, into one or both classes of the training data and are specifically designed to cause a significant increase in false positives or false negatives for the resulting classifier. They conclude that the generation of classifiers for adversarial environments should take into account the fact that training data is controlled by an adversarial source in order to ensure the production of accurate classifiers.

Barreno et al. (2006) explore possible adversarial attacks on machine learning algorithms from multiple perspectives. They present a taxonomy of different types of attacks on machine learning systems. An attack is *causative* if it targets the training data, and is *exploratory* if it aims to discover information through, for example, offline analysis. An attack is *targeted* if it focuses on a small set of points, and is *indiscriminate* if it targets a general class of points. An *integrity* attack leads to false negatives, and an *availability* attack aims to cause (machine learning) system dysfunction by generating many false negatives and false positives. They also discuss several potential defences against those attacks, and give a lower bound on the adversary's effort in attacking a naïve learning algorithm.

A practical example of adversarial learning is learning in the presence of the good word attack. Lowd & Meek (2005b) present and evaluate several variations of this type of attack on spam filters. They demonstrate two different ways to carry out the attack: passively and actively. Active good word attacks use feedback obtained by sending test messages to a spam filter in order to determine which words are "good". The active attacks were found to be more effective than the passive attacks; however, active attacks are generally more difficult than passive attacks because they require user-level access to the spam filter, which is not always possible. Passive good word attacks, on the other hand, do not involve any feedback from the spam filter, but rather, guesses are made as to which words are considered good. Three common ways for passively choosing good words are identified. First, *dictionary attacks* involve selecting random words from a large collection of words, such as a dictionary. In testing, this method did not prove to be effective; in fact, it actually increased the chances that the e-mail would be classified as spam. Next, *frequent word attacks* involve the selection of words that occur most often in legitimate messages, such as news articles. This method was more effective than the previous one, but it still required as many as 1,000 good words to be added to the original message. Finally, *frequency ratio attacks* involve the selection of words that occur very often in legitimate messages but not in spam messages. The authors' tests showed that this technique was quite effective, resulting in the average spam message being passed off as legitimate by adding as few as 150 good words to it. Preliminary results were also presented that suggested that frequent retraining on attacked messages may help reduce the effect of good word attacks on spam filters.

Webb et al. (2005) also examined the effectiveness of good word attacks on statistical spam filters. They present a "large-scale evaluation" of the effectiveness of the attack on four spam filters: naïve Bayes, support vector machine (SVM), LogitBoost, and Spam-Probe. Their experiments were performed on a large e-mail corpus consisting of around a million spam and ham messages, which they formed by combining several public and private corpora. Such a large and diverse corpus more closely simulates the environment of a server-level spam filter than a client-level filter. The experimental results show that, on normal e-mail, i.e., e-mail that has not been modified with good words, each of the filters is able to attain an accuracy as high as 98%. When testing on "camouflaged messages", however, the accuracies of the filters drop to between 50% and 75%. In their experiments, spam e-mails were camouflaged by combining them with portions of legitimate messages. They experimented with camouflaged messages containing twice as much spam content as legitimate content, and vice versa. They also proposed and demonstrated a possible solution to the attack. By training on a collection of e-mails consisting of half normal and half camouflaged messages, and treating all camouflaged messages as spam, they were able to improve the accuracy of the filters when classifying camouflaged messages.

Our counterattack strategy against good word attacks is inspired by work in the field of multiple instance (MI) learning. The concept of MI learning was initially proposed by Dietterich et al. (1997) for predicting drug activities. The challenge of identifying a drug molecule that binds strongly to a target protein is that a drug molecule can have multiple conformations, or shapes. A molecule is positive if at least one of its conformations binds tightly to the target, and negative if none of its conformations bind well to the target. The problem was tackled with an MI model that aims to learn axis-parallel rectangles (APR). Later, learning APR in the multiple instance setting was further studied and proved to be NP-complete by several other researchers in the PAC-learning framework (Auer, 1997; Long and Tan, 1998; Blum and Kalai, 1998).

Several probabilistic models: diverse density (DD) (Maron & Lozano-Pérez, 1998) and its variation EM-DD (Zhang & Goldman, 2002), and multiple instance logistic regression (MILR) (Ray & Craven, 2005), employ a maximum likelihood estimation to solve problems in the MI domain. The original DD algorithm searches for the target concept by finding an area in the feature space with maximum diverse density, i.e., an area with a high density of positive points and a low density of negative points. The diverse density at a point in the feature space is defined to measure probabilistically how many different positive bags have instances near that point, and how far the negative instances are from that point. EM-DD combines EM with the DD algorithm to reduce the multiple instance learning problem to a single-instance setting. The algorithm uses EM to estimate the instance in each bag which is most likely to be the one responsible for the label of the bag. The MILR algorithm presented by Ray & Craven (2005) is designed to learn linear models in a multiple instance setting. Logistic regression is used to model the posterior probability of the label of each instance in a bag, and the bag level posterior probability is estimated by using softmax to combine the posterior probabilities over the instances of the bag. Similar approaches with different combining functions are presented by Xu & Frank (2004).

Many single-instance learning algorithms have been adapted to solve the multiple instance learning problem. For example, Wang & Zucker (2000) propose the lazy MI learning algorithms, namely Baysian-kNN and citation-kNN, which solve the multiple instance learning problem by using the Hausdorff distance to measure the distance between two bags of points in the feature space. Chevaleyre & Zucker (2001) propose the multi-instance decision tree ID3-MI and decision rule learner RIPPER-MI by defining a new multiple instance entropy function and a multiple instance coverage function. Other algorithms that have been adapted to multiple instance learning include the neural network MI-NN (Ramon & Raedt, 2000), DD-SVM (Chen & Wang, 2004), MI-SVM and mi-SVM (Andrews et al., 2003), multi-instance kernels (Gärtner et al., 2002), MI-Ensemble (Zhou & Zhang, 2003), and MI-Boosting (Xu & Frank, 2004).

In this chapter, we demonstrate that a counterattack strategy against good word attacks, developed in the framework of multiple instance learning, can be very effective, provided that a single instance can be properly transformed into a bag of instances. We also explore several possible ways to transform e-mails into bags of instances. Our experiments also verify earlier observations, discussed in other works (Lowd & Meek, 2005b; Webb et al., 2005), that retraining on e-mails modified during adversarial attacks may improve the performance of the filters against the attack.

3. Problem definition

Consider a standard supervised learning problem with a set of training data $D = \{<X_1, Y_1>, ..., <X_m, Y_m>\}$, where X_i is an instance represented as a single feature vector, $Y_i = C(X_i)$ is the target value of X_i, where *C* is the target function. Normally, the task is to learn *C* given *D*. The learning task becomes more difficult when there are adversaries who could alter some instance so that $X_i \rightarrow X_i^{'}$ and cause $Y_i \rightarrow Y_i^{'}$, where $Y_i \neq Y_i^{'}$. Let ΔX_i be the difference between X_i and $X_i^{'}$, i.e., $X_i^{'} = X_i + \Delta X_i$. In the case of spam filtering, an adversary can modify spam e-mails by injecting them with good words. So, ΔX_i represents a set of good words added to a spam message by the spammer. There are two cases that need to be studied separately:

1. the filter is trained on normal e-mails, i.e., e-mails that have not been injected with good words, and tested on e-mails which have been injected with good words;
2. both the training and testing sets contain e-mails injected with good words.

In the first case, the classifier is trained on a clean training set. Predictions made for the altered test instances are highly unreliable. In the second case, the classifier may capture some adversarial patterns as long as the adversaries consistently follow a particular pattern. In both cases, the problem becomes trivial if we know exactly how the instances are altered; we could recover the original data and solve the problem as if no instances were altered by the adversary. In reality, knowing exactly how the instances are altered is impossible. Instead, we seek to approximately separate X_i and ΔX_i and treat them as separate instances in a bag. We then apply multiple instance learning to learn a hypothesis defined over a set of bags.

4. Multiple instance bag creation

We now formulate the spam filtering problem as a multiple instance binary classification problem in the context of adversarial attacks. Note that the adversary is only interested in altering positive instances, i.e., spam, by injecting sets of good words that are commonly encountered in negative instances, i.e., legitimate e-mails, or ham. We propose four different approaches to creating multiple instance bags from e-mails. We call them split-half (split-H), split-term (split-T), split-projection (split-P), and split-subtraction (split-S). We will now discuss each of these splitting methods, in turn.

4.1 Split-H

In our first splitting method, *split-half*, we split every e-mail right down the middle into approximately equal halves. Formally, let $B = \{B_1, ..., B_i, ..., B_m\}$ be a set of bags (e-mails), where $B_i = \{X_{i1}, X_{i2}\}$ is the i^{th} bag, X_{i1} and X_{i2} are the two instances in the i^{th} bag, created from the upper half and the lower half of the e-mail respectively. This splitting approach is reasonable in practice because spammers usually append a section of good words to either the beginning or the end of an e-mail to ensure the legibility of the spam message.

This splitting method, because it relies on the physical positioning of words in an e-mail, could potentially be circumvented by the spammer. There are two obvious ways to do this.

One is to create a visual pattern with good words so that the original spam message is still legible after the attack, but the spam is fragmented in such a way that "spammy" words are well separated and become less indicative of spam in the presence of a number of good words. The following example demonstrates how this idea can work effectively to circumvent this splitting method.

From: foo@internet.org
To: foo-foo@email.org
Subject: meeting agenda

good words	…	**low**	…	good words
good words	…	**mortgage**	…	good words
good words	…	**rate**	…	good words

The second way to defeat the split-H method is to append a very large block of good words to an e-mail so that after the split, good words outweigh spam-indicative words in all instances. The next three splitting methods do not rely on the positions of the words and thus do not suffer from this vulnerability.

4.2 Split-T

The second splitting method, *split-term* (split-T), partitions a message into three groups of words (terms) depending on whether the word is an indicator of spam, an indicator of ham, or neutral, i.e., $B_i = \{X_{is}, X_{in}, X_{ih}\}$, where X_{is} is the spam-likely instance, X_{in} is the neutral instance, and X_{ih} is the ham-likely instance in bag B_i. The instance to which each word is assigned is based on a weight generated for it during preprocessing. These weights are calculated using word frequencies obtained from the spam and legitimate messages in the training corpus. More specifically, the weight of a term W is given as follows:

$$weight(W) = \frac{p(W \mid D_s)}{p(W \mid D_s) + p(W \mid D_h)},$$

where D_s and D_h are the spam and ham e-mails in the training set respectively. When splitting an e-mail into instances we used two threshold values, $thresh_s$ and $thresh_\ell$, to determine which instance (spam-likely, ham-likely, or neutral) each word in the e-mail should be assigned to. We considered any word with a weight greater than $thresh_s$ to be spammy, any word with a weight less than $thresh_\ell$ to be legitimate, and any word with a weight in between to be neutral. Given each training set, $thresh_s$ was selected such that some fraction, for example 20%, of the terms chosen during attribute selection (discussed in Section 6.1) would have a weight greater than or equal to it. $thresh_\ell$ was selected so that some other fraction, for example 50%, of the terms would have a weight less than or equal to it. Reasonable fractions of terms and threshold values can be determined by using cross validation on training e-mails.

4.3 Split-P

The third splitting method, *split-projection* (split-P), transforms each message into a bag of two instances by projecting the message vector onto the spam and ham prototype vectors. The prototype vectors are computed using all the spam and ham messages in the training set. If we view the spam and ham messages in the training set as two clusters, then the prototypes are essentially the centroid of the two clusters. More specifically, let C_s be the set of e-mails that are spam and C_l be the set of e-mails that are legitimate. The prototypes are computed using Rocchio's algorithm (Rocchio Jr., 1971) as follows:

$$P_s=\beta\cdot 1/|C_s|\cdot \sum_{i=1}^{|C_s|} C_{s_i}-\gamma\cdot 1/|C_\ell|\cdot \sum_{i=1}^{|C_\ell|} C_{\ell_i}$$

$$P_\ell=\beta\cdot 1/|C_\ell|\cdot \sum_{i=1}^{|C_\ell|} C_{\ell_i}-\gamma\cdot 1/|C_s|\cdot \sum_{i=1}^{|C_s|} C_{s_i}$$

where C_{s_i} is the i^{th} spam message in C_s and $C_{\ell i}$ is the i^{th} ham message in C_ℓ, β is a fixed constant suggested to be 16 and γ is a fixed constant suggested to be 4. Given a message *M*, two new instances, M_S and M_ℓ, are formed by projecting *M* onto P_s and P_ℓ:

$$M_s = \frac{M \cdot P_s}{|P_s|^2} P_s$$

$$M_\ell = \frac{M \cdot P_\ell}{|P_\ell|^2} P_\ell$$

The rationale of this splitting approach rests on the assumption that a message is close to the spam prototype in terms of cosine similarity if it is indeed spam, and a ham message is close to the ham prototype.

4.3 Split-S

The last splitting method, *split-subtraction* (split-S), like the former, uses prototype (centroid) vectors. In this method, however, the ham and spam prototypes are calculated by averaging the corresponding attribute values of all of the ham and spam e-mails, respectively.

$$P_s=1/|C_s|\cdot \sum_{i=1}^{|C_s|} C_{s_i}$$

$$P_\ell=1/|C_\ell|\cdot \sum_{i=1}^{|C_\ell|} C_{\ell_i}$$

where C_s is a set of spam and C_{s_i} is the i^{th} spam message in C_s ; C_ℓ is a set of ham, and C_{ℓ_i} is the i^{th} ham message in C_ℓ . A message can then be transformed from a single instance attribute vector *M* into a bag of two instances by subtracting corresponding attribute values in the single instance vector from the ham prototype and the spam prototype, yielding a legitimate instance $M_\ell = M - P_\ell$ and a spam instance $M_s = M - P_s$, respectively (Zhang & Zhou, 2007).

Now that we have devised several techniques for creating multiple instance bags from e-mail messages, we can transform the standard supervised learning problem of spam filtering into a multiple instance learning problem under the standard MI assumption. For this research, we adopt the multiple instance logistic regression (MILR) model to train a spam filter that is more robust to adversarial good word attacks than traditional spam filters based on single instance models. We chose to use the MILR classifier over other MI classifiers mainly because its single instance counter-part, logistic regression (LR), which has been shown to be very effective in the spam filtering domain (Yih et al., 2006), appeared to be the best among the single instance learners considered in our experiments. The next section outlines the multiple instance logistic regression learning model.

5. Multiple instance logistic regression

Given a set of training bags $B = \{< B_1, Y_1 >, ..., < B_i, Y_i >, ..., < B_m, Y_m >\}$, let $Pr(Y_i = 1 | B_i)$ be the probability that the i^{th} bag is positive, and $Pr(Y_i = 0 | B_i)$ be the probability that it is negative. Here Y_i is a dichotomous outcome of the i^{th} bag (e.g., spam or legitimate). The bag-level binomial log-likelihood function is:

$$L = \sum_{i=1}^{m} [Y_i \log Pr(Y_i = 1 | B_i) + (1 - Y_i) \log Pr(Y_i = 0 | B_i)].$$

In a single instance setting where logistic regression is used, given an example X_i, we model the expected value of the dichotomous outcome of X_i with a sigmoidal response function, i.e., $Pr(Y_i = 1 | X_i) = \exp(p \cdot X_i + b)/(1 + \exp(p \cdot X_i + b))$, then estimate the parameters $\boldsymbol{p}$ and b that maximize the log-likelihood function. In a multiple instance setting, we do not have direct measure of bag-level probabilities in the log-likelihood function. Instead, we estimate the instance-level class probabilities $Pr(Y_{ij} = 1 | X_{ij})$ with a sigmoidal response function as follows:

$$Pr(Y_{ij} = 1 | X_{ij}) = \frac{\exp(p \cdot X_{ij} + b)}{1 + \exp(p \cdot X_{ij} + b)},$$

where X_{ij} is the j^{th} instance in the i^{th} bag, and $\boldsymbol{p}$ and b are the parameters that need to be estimated. Thus, $Pr(Y_i = 0 | B_i)$ with instance-level class probabilities can be computed as follows:

$$Pr(Y_{ij} = 0 | X_{ij}) = 1 - Pr(Y_{ij} = 1 | X_{ij}) = \frac{1}{1 + \exp(p \cdot X_{ij} + b)}.$$

Now we can compute the probability that a bag is negative based on the MI assumption that a bag is negative if and only if every instance in the bag is negative:

$$Pr(Y_i = 0 \mid B_i) = \prod_{j=1}^{n} Pr(Y_{ij} = 0 \mid X_{ij})$$

$$= \exp(-\sum_{j=1}^{n} (\log(1 + \exp(p \cdot X_{ij} + b)))),$$

where n is the number of instances in the i^{th} bag. Thus the probability

$$Pr(Y_i = 1 \mid B_i) = 1 - Pr(Y_i = 0 \mid B_i)$$

estimates how likely a bag is positive if at least one instance in the bag is positive. In our case, given a set of e-mails for training, X_{ij} is a vector of the frequency count (or other variations such as a *tf-idf* weight) of unique terms in each e-mail. We can apply maximum likelihood estimation (MLE) to maximize the bag-level log-likelihood function, and estimate the parameters $\boldsymbol{p}$ and b that maximize the probability of observing the bags in B.

6. Experimental setup

We evaluated our multiple instance learning counterattack strategy on e-mails from the 2006 TREC Public Spam Corpus (Cormack & Lynam, 2006). Good word attacks were simulated by generating a list of good words from the corpus and injecting them into spam messages in the training and/or test data sets. We compared our counterattack strategy, using the multiple instance logistic regression model and the four splitting methods introduced above, to its single instance learning counterpart—logistic regression (LR)—and to the support vector machine (SVM) and the multinomial naïve Bayes (MNB) classifiers.

6.1 Experimental data

Our experimental data consists of 36,674 spam and legitimate e-mail messages from the 2006 TREC spam corpus. We preprocessed the entire corpus by stripping HTML and non-textual parts and applying stemming and stop-list to all terms. The **to**, **from**, **cc**, **subject**, and **received** headers were retained, while the rest of the headers were stripped. Messages that had an empty body after preprocessing were discarded. Tokenization was done by splitting on nonalphanumeric characters. We did not take any measures to counter obfuscated words in the spam messages. Given that there are a large number of possibilities to disguise a word, most content-based spam filters will not be able to deobfuscate the text of a message efficiently (Carpinter & Hunt, 2006). Recently, an efficient complementary filter (Lee & Ng, 2005) has been demonstrated to be able to effectively deobfuscate text with high accuracy. In practice, this type of technique could be used during preprocessing.

For our experiments we sorted the e-mails in the corpus chronologically by receiving date and evenly divided them into 11 subsets $\{D_1,...,D_{11}\}$. In other words, the messages in subset n come chronologically before the messages in subset n+1. Experiments were run in an on-line fashion, i.e., training on subset n and testing on subset n+1. Each subset contains approximately 3300 messages. The percentage of spam messages in each subset varies as in the operational setting (see Figure 1). We used the Multiple Instance Learning Tool Kit (MILK) (Xu, 2003) implementation of MILR and the Weka 3.4.7 (Witten and Frank, 2000) implementations of LR, SVM and multinomial naïve Bayes, in our experiments. We reduced the feature space to the top 500 words ranked using information gain. Retaining 500 features appeared to be the best compromise among the classifiers in terms of improved efficiency and impaired performance. Attribute values for each message are calculated using the *tf-idf* (term frequency inverse document frequency) weighting scheme.

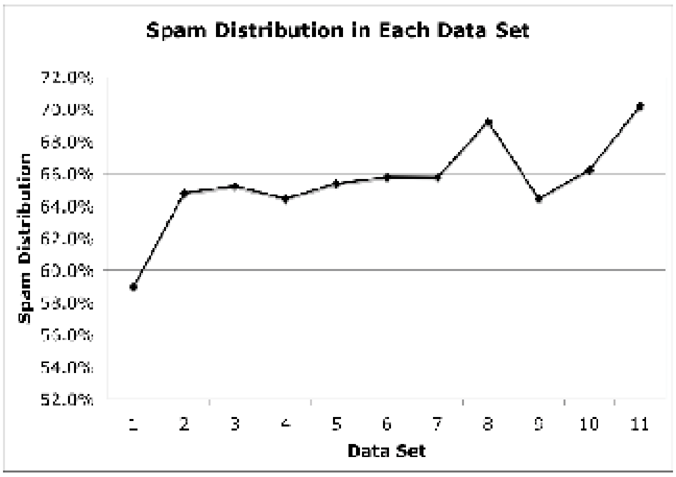


Fig. 1. Percentage of e-mails in each data set that are spam.

6.2 Good word list

To simulate the good word attacks, we generated a list of good words using all the messages in the TREC spam corpus. We ranked each word according to the ratio of its frequency in the legitimate messages over its frequency in the spam messages. We then selected the top 1,000 words from the ranking to use as our good word list. Generating the good word list in this manner has an important implication. Since the list was generated from the entire corpus rather than from the subset of messages used to train the classifiers, and since we represent e-mails using a feature vector of 500 features, some of the words in the list will not have an effect on the classification of messages that they are injected into. Such a list is more representative of the kind of list a spammer would be able to produce in practice, since the spammer would have no way of knowing the exact features used by the target filter. We noticed that in our experiments, only about 10% of the injected good words were actually retained in the feature vector, yet they had a significant impact on the classification.

7. Experimental results

We now present the results of two experiments in which we evaluate the effectiveness of our proposed multiple instance counterattack strategy. In the first experiment, we train all of the classifiers on normal e-mail (i.e., e-mail that has not been injected with good words) and then test them on e-mail that has been injected with good words. In the second experiment we train on both normal and attacked e-mails to observe how doing so affects classification of both normal and attacked e-mails.

7.1 Experiment 1: attacking the test set

In this experiment, we tested the ability of the MILR algorithm, using the four splitting methods introduced above, to classify e-mail injected with good words. We also tested the single instance logistic regression (LR), support vector machine (SVM) and multinomial naïve Bayes (MNB) classifiers for comparison. The classifiers were each trained and tested on the eleven chronologically sorted data sets in an on-line fashion. That is, all of the classifiers were trained on the same unaltered data set D_n, and then tested on the data set D_{n+1}, for n=1...10. Fifteen variations of each test set were created to test the susceptibility of the classifiers to good word attacks of varying strength. The first version of each test set was left unmodified, i.e., no good words were injected. Half of the spam messages (selected at random) in each of the remaining 14 variations of each test set were injected with some quantity of random good words from our good word list, beginning with 10 words. With each successive version of the test set, the quantity of good words injected into half of the spam messages was increased: first in increments of 10 words, up to 50, and then in increments of 50 words up to 500. The injected words were randomly selected, without replacement, from our good word list on a message by message basis. We chose to inject good words into only half of the messages in each test set because, in practice, spam messages injected with good words account for only a subset of the spam e-mails encountered by a given filter. The precision and recall values on each version of the test set for all 10 test sets were averaged and recorded for each classifier. In our results, we use "MILRH", "MILRT", "MILRP" and "MILRS" where split-H, split-T, split-P and split-S were used with MILR, respectively.

Figures 2 and 3 show how the average precision and average recall, respectively, of each classifier is affected as the good word attack increases in strength (that is, the quantity of good words injected into the spam e-mails in the test set increases).

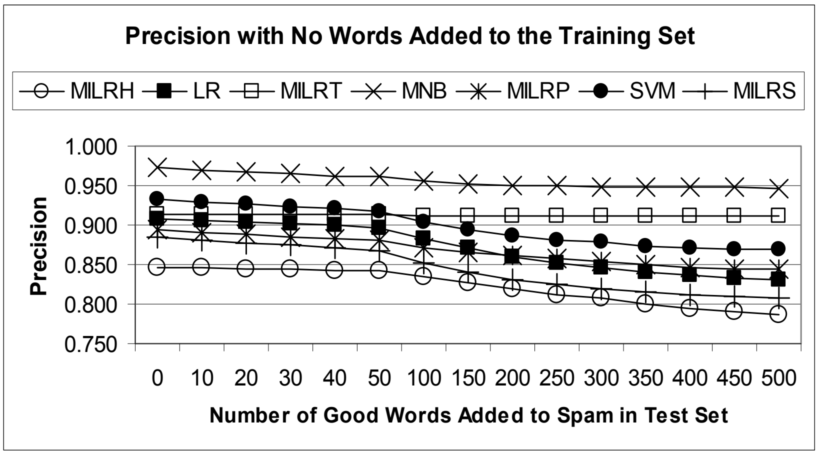


Fig 2. Change in average precision as good words are injected into the test Set.

From the results we can see that, with the exception of MILRT, each classifiers' ability to correctly identify spam e-mail was significantly reduced as a result of the simulated good word attack. Of all the classifiers MILRT was most resilient to the attack, dropping by only 0.3% (from 0.914 to 0.911) in average precision and by only 2% (from 0.901 to 0.883) in average recall after 500 good words had been injected into the test set. MILRH and MILRP stood up better to the attack than the single instance classifiers and the MILRS classifier, but the attack still had a very noticeable effect on their ability to classify spam, reducing the average recall of MILRH by 27.1% (from 0.980 to 0.714) and the average recall of MILRP by 29.2% (from 0.957 to 0.678). The average precision values of MILRH and MILRP dropped by 7% (from 0.846 to 0.787) and by 5.7% (from 0.895 to 0.844), respectively. Of the single instance classifiers, LR was the most resilient; however, the attack still had a very significant effect on its ability to classify spam, reducing its average recall by 42.8% (from 0.966 to 0.553) and its average precision by 8.6% (from 0.909 to 0.831). The average recall of MNB and SVM dropped by 47.4% (from 0.914 to 0.481) and 46.2% (from 0.932 to 0.501), respectively. Their average precision values dropped by 2.7% (from 0.973 to 0.947) and by 6.9% (from 0.932 to 0.868), respectively. MILRS turned out to be nearly as vulnerable to the attack as the single instance classifiers, dropping by 37.8% (from 0.961 to 0.598) in average recall and by 8.7% (from 0.885 to 0.808) in average precision.

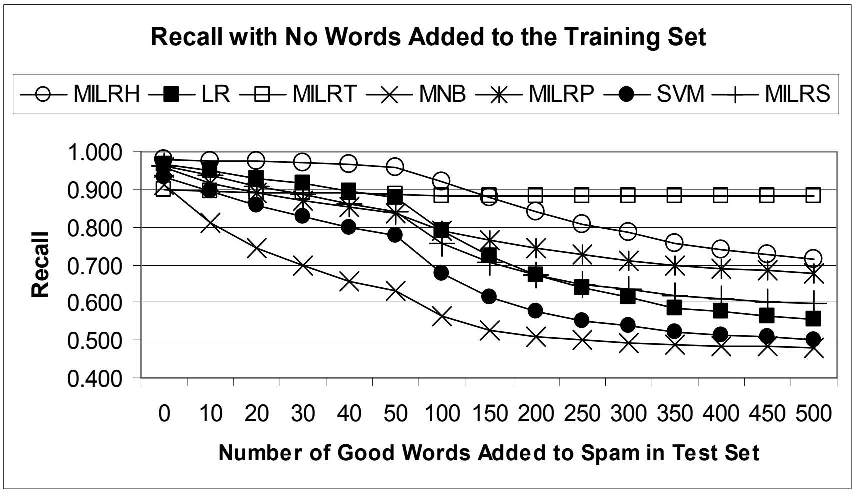


Fig 3. Change in average recall as good words are injected into the test set.

One thing that is clear from these results is that the effectiveness of our multiple instance counterattack strategy is very much dependent on the specific technique used to split e-mails into multiple instance bags. The success of the split-term method (MILRT) is due to the fact that the classifier is able to consider both spammy and legitimate terms independently, since they are placed into separate instances in the bag created from an e-mail. Under the multiple instance assumption, if at least one instance in a bag is spammy, the entire bag is labeled as spammy. When good words are injected into a spam message they end up in the legitimate instance of the bag and have no effect on the spammy instance; thus the bag still contains a spammy instance and is classified correctly as spam.

7.2 Experiment 2: training on attacked spam messages

In the second experiment, our goal was to observe the effect that training on messages injected with good words has on the susceptibility of the classifiers to attacks on the test set. As in the previous experiment, we tested each of the classifiers on the eleven chronologically sorted data sets in an on-line fashion. This time, however, in addition to creating 15 versions of the test set injected with increasing quantities of good words, we also created 5 versions of the training set. We injected 10 good words into half of the spam messages (selected at random) in the first version of the training set and then increased the number of injected good words by 10 for each subsequent version, up to 50 good words for the fifth version. We also tried injecting larger numbers of good words, but after exceeding 50 words, the additional effect was minimal; therefore, those results are not shown here. For each version of the training set we tested the classifiers on the 15 versions of the corresponding test set. As before, good words were selected from our good word list randomly and without replacement on a message by message basis. For all ten tests, the precision and recall values on each version of the test set were averaged and recorded, separately for each of the 5 versions of the training set. In the interest of space, the results of this experiment are presented in terms of average F-measure that is the harmonic mean of the precision and recall values, and only the results of the experiments when 10, 30 and 50 good words are added to some spam in the training set are presented. Figures 4-6 show the average F-measure of each of the classifiers when 10, 30 and 50 good words are injected into half of the spam messages in the training set, respectively.

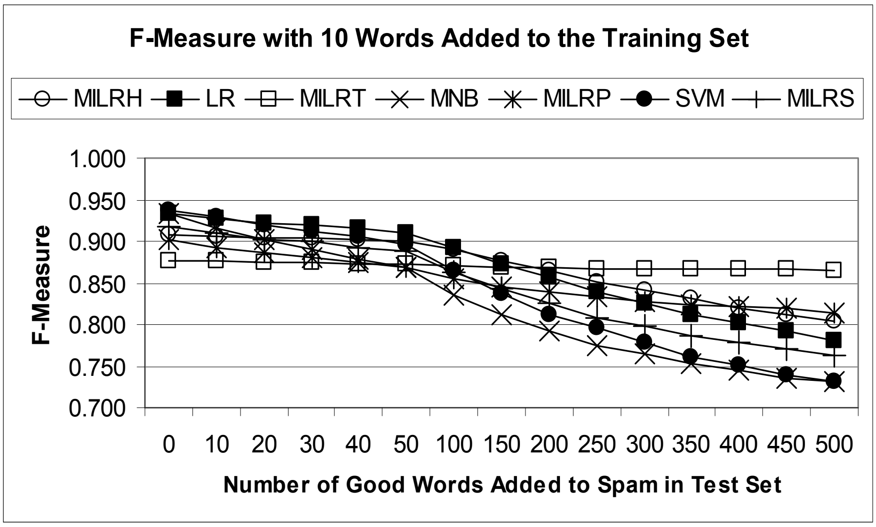


Fig 4. Change in average F-measure as good words are injected into the test set. 10 good words were also injected into the training set.

From these results we can see that injecting just 10 good words into half of the spam messages in the training set appeared to lessen the effect of the good word attack for almost all of the classifiers (see Fig. 4). Further increasing the number of good words injected into the training set continued to lessen the effect of the attack for all of the classifiers (see Fig. 5). After 30 good words had been injected into the training set, the presence of good words in the test messages actually began to increase the likelihood that such messages would be correctly classified as spam (see Fig. 6). These results confirm the observations of several other researchers (Lowd and Meek, 2005b; Webb et al., 2005) that retraining on normal and attacked e-mails may help to counter the effects of the good word attack. However, it is important to realize that this would only work in cases where the attacked messages being classified contained the same good words as the attacked messages that the spam filter was trained on. One of the major advantages of our proposed multiple instance strategy is that the spam filter need not be trained on attacked messages in order to be effective against attacks and further, that frequent retraining on attacked messages is not necessary for the strategy to maintain its effectiveness.

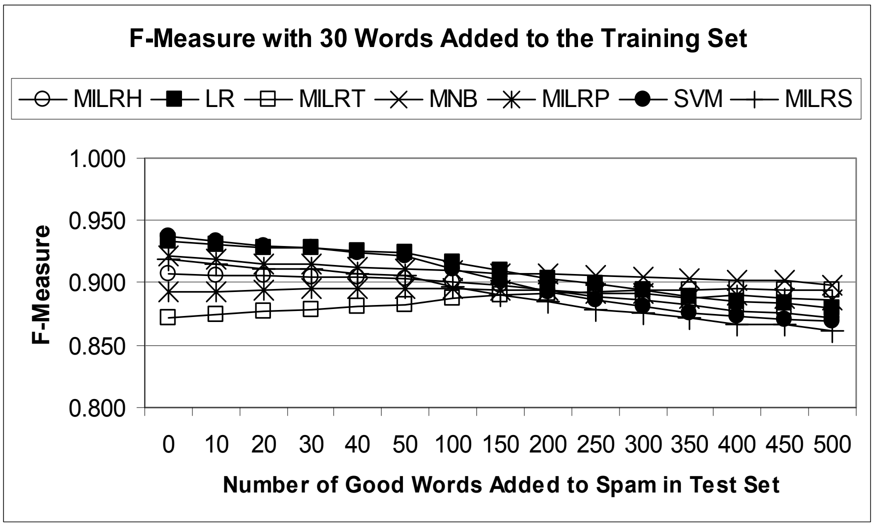


Fig 5. Change in average F-measure as good words are injected into the test set. 30 good words were also injected into the training set.

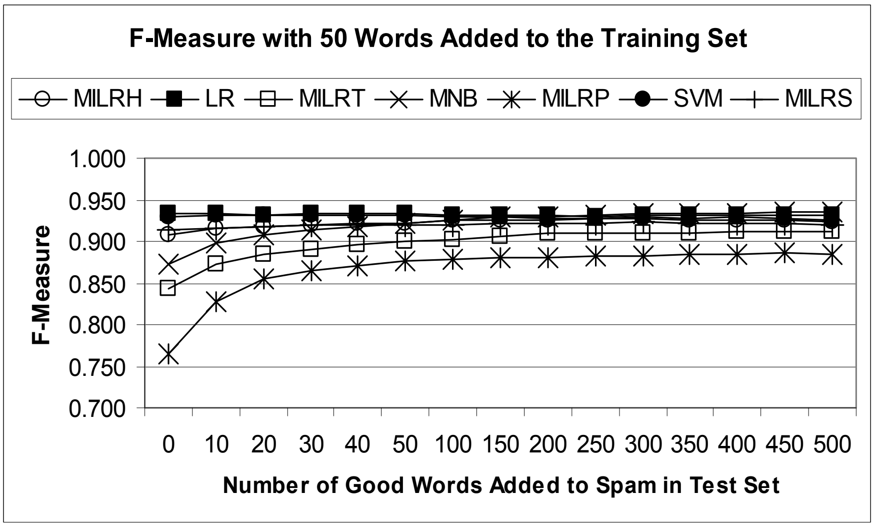


Fig 6. Change in average F-measure as good words are injected into the test set. 50 good words were also injected into the training set.

8. Conclusions and future work

A multiple instance learning counterattack strategy for combating adversarial good word attacks on statistical spam filters has been presented. In the proposed strategy, e-mails are treated as multiple instance bags and a logistic model at the instance level is learned indirectly by maximizing the bag-level binomial log-likelihood function. The proposed counterattack strategy has been demonstrated on good word attacks of varying strength and has been shown to be effective. Additionally, we have confirmed earlier reports that

retraining on attacked as well as normal e-mails may strengthen a spam filter against good word attacks. One of the advantages of our proposed strategy, as demonstrated by our experiments, is that it is effective even when trained on normal e-mail and that frequent retraining on attacked messages is not necessary to maintain that effectiveness. We presented several possible methods for creating multiple instance bags from e-mails. As was observed from our experimental results, the splitting method used ultimately determines how well the strategy performs. The splitting methods we presented here work fairly well, especially the split-term method, but there are possibly other, perhaps better, methods that could be used. We plan to investigate other possible splitting methods in the future, and investigate the vulnerability of each splitting method in adversarial environments. In our experiments, we simulated the real operational environment where the adversary does not have a complete knowledge of the training data. We plan to investigate the effectiveness of our proposed counterattack strategy in the extreme cases where we assume the adversary knows the make-up of the training examples.
Since it is an arms race between spammers and filter designers, we also plan to make our MI strategy adaptive as new spam techniques are devised, and on-line as the concept of spam drifts over time. In addition, we plan to investigate the possibility of extending the proposed multiple instance learning strategy to handle similar adversarial attacks in other domains.

9. References

S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple instance learning. In *NIPS 15*, pages 561–568. MIT Press, 2003.

P. Auer. On learning from multi-instance examples: Empirical evaluation of a theoretical approach. In *Proceedings of the 14th International Conference on Machine Learning*, pages 21–29, San Francisco, CA, 1997. Morgan Kaufmann.

M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can machine learning be secure? In ASIACCS '06: *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-272-0. doi: http://doi.acm.org/10.1145/1128817.1128824.

A. Blum and A. Kalai. A note on learning from multiple-instance examples. *Machine Learning*, 30(1):23–30, 1998.

J. Carpinter and R. Hunt. Tightening the net: A review of current and next generation spam filtering tools. *Computers and Security*, 25(8):566–578, 2006.

Y. Chen and J.Z. Wang. Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, 5:913–939, 2004.

Y. Chevaleyre and J.D. Zucker. Solving multiple-instance and multiple-part learning problems with decision trees and rule sets. application to the mutagenesis problem. In *Proceedings of the 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pages 204–214, 2001.

G. V. Cormack and T. R. Lynam. Spam track guidelines — TREC 2005-2007. http://plg.uwaterloo.ca / gvcormac/treccorpus06/, 2006.

N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108. ACM Press, 2004.

T.G. Dietterich, R.H. Lathrop, and T. Lozano-Pérez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence Journal*, 89(1-2):31–71, 1997.

T. Gärtner, P. Flach, A. Kowalczyk, and A. Smola. Multi-instance kernels. In *Proceedings Of the 19th International Conference on Machine Learning*, pages 179–186, San Francisco, CA, 2002. Morgan Kaufmann.

R. Jennings. The global economic impact of spam. Technical report, Ferris Research, 2005.

J.Z. Kolter and M.A. Maloof. Using additive expert ensembles to cope with concept drift. In *Proceedings of the Twenty-second International Conference on Machine Learning*, pages 449–456, New York, NY, 2005. ACM Press.

H. Lee and A. Ng. Spam deobfuscation using a hidden markov model. In *Proceedings of the Second Conference on Email and Anti-Spam*, 2005.

P. Long and L. Tan. Pac learning axis-aligned rectangles with respect to product distribution from multiple-instance examples. *Machine Learning*, 30(1):7–21, 1998.

D. Lowd and C. Meek. Adversarial learning. In *Proceedings of the 2005 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 641 647. ACM Press, 2005a.

D. Lowd and C. Meek. Good word attacks on statistical spam filters. In *Proceedings of the 2nd Conference on Email and Anti-Spam*, 2005b.

O. Maron and T. Lozano-P´erez. A framework for multiple-instance learning. *Advances in Neural Information Processing Systems*, 10:570–576, 1998.

J. Newsome, B. Karp, and D. Song. Paragraph: Thwarting signature learning by training maliciously. In *Recent Advances in Intrusion Detection: 9th International Symposium (RAID)*, pages 81–105, 2006.

J. Ramon and L.D. Raedt. Multi instance neural networks. In *Proceedings of ICML-2000 workshop on Attribute-Value and Relational Learning*, 2000.

S. Ray and M. Craven. Supervised versus multiple instance learning: An empirical comparison. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 697–704, New York, NY, 2005. ACM Press.

J. Rocchio Jr. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 68–73. Prentice Hall, 1971.

J. Wang and J.D. Zucker. Solving the multiple-instance learning problem: A lazy learning approach. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1119–1125, San Francisco, CA, 2000. Morgan Kaufmann.

S. Webb, S. Chitti, and C. Pu. An experimental evaluation of spam filter performance and robustness against attack. In *The 1st International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 19–21, 2005.

I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, CA, USA, 2000.

X. Xu. Statistical learning in multiple instance problems. Master's thesis, University of Waikato, 2003.

X. Xu and E. Frank. Logistic regression and boosting for labeled bags of instances. In *Proceedings of the Pacific-Asian Conference on Knowledge discovery and data mining*. Springer-Verlag, 2004.

W. Yih, J. Goodman, and G. Hulten. Learning at low false positive rates. In *Proceedings of the Third Conference on Email and Anti-Spam*, 2006.

Q. Zhang and S. Goldman. Em-dd: An improved multiple-instance learning technique. In *Proceedings of the 2001 Neural Information Processing Systems (NIPS) Conference*, pages 1073–1080, Cambridge, MA, 2002. MIT Press.

M. L. Zhang and Z. H. Zhou. 2007. Multi-label learning by instance differentiation. In The 22nd AAAI Conference on Artificial Intelligence (AAAI'07), pages 669–674, Vancouver, Canada.

Z.H. Zhou and M.L. Zhang. Ensembles of multi-instance learners. In *ECML-03, 15th European Conference on Machine Learning*, pages 492–502, 2003.

Edited by Paula Fritzsche

This book offers in 27 chapters a collection of all the technical aspects of specifying, developing, and evaluating the theoretical underpinnings and applied mechanisms of AI tools. Topics covered include neural networks, fuzzy controls, decision trees, rule-based systems, data mining, genetic algorithm and agent systems, among many others. The goal of this book is to show some potential applications and give a partial picture of the current state-of-the-art of AI. Also, it is useful to inspire some future research ideas by identifying potential research directions. It is dedicated to students, researchers and practitioners in this area or in related fields.

IntechOpen

ISBN 978-953-51-5794-6

9 789535 157946