# Advances in Artificial Intelligence Theory

# Research on Computing Science




**Research on Computing Science** es una publicación trimestral, de circulación internacional, editada por el Centro de Investigación en Computación del IPN, para dar a conocer los avances de investigación científica y desarrollo tecnológico de la comunidad científica internacional. **Volumen 16** Noviembre, 2005. Tiraje: 500 ejemplares. *Certificado de Reserva de Derechos al Uso Exclusivo del Título* No. 04-2004-062613250000-102, expedido por el Instituto Nacional de Derecho de Autor. *Certificado de Licitud de Título* No. 12897, *Certificado de licitud de Contenido* No. 10470, expedidos por la Comisión Calificadora de Publicaciones y Revistas Ilustradas. El contenido de los artículos es responsabilidad exclusiva de sus respectivos autores. Queda prohibida la reproducción total o parcial, por cualquier medio, sin el permiso expreso del editor, excepto para uso personal o de estudio haciendo cita explícita en la primera página de cada documento. Se usó la imagen obtenida de la siguiente dirección, para el diseño de la portada: www.absolutewallpapers.com/wallpapers/3dwallpapers/fractal/fractal_2.jpg. Impreso en la Ciudad de México, en los Talleres Gráficos del IPN – Dirección de Publicaciones, Tres Guerras 27, Centro Histórico, México, D.F. Distribuida por el Centro de Investigación en Computación, Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othón de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738, México, D.F. Tel. 57 29 60 00, ext. 56571.

**Editor Responsable:** *Juan Humberto Sossa Azuela*, *RFC SOAJ560723*

***Research on Computing Science*** *is published by the Center for Computing Research of IPN.* ***Volume 16****, November, 2005. Printing 500. The authors are responsible for the contents of their articles.*  *Printed in Mexico City, November, 2005, in the IPN Graphic Workshop – Publication Office.*


Volume 16

Volumen 16

# Advances in Artificial Intelligence Theory

**Volume Editors:**
**Editores del Volumen**

*Alexander Gelbukh*
*Raúl Monroy*

Instituto Politécnico Nacional
Centro de Investigación en Computación
México 2005

**ISSN: 1665-9899**



Instituto Politécnico Nacional (IPN)
Centro de Investigación en Computación (CIC)
Av. Juan de Dios Bátiz s/n esq. M. Othón de Mendizábal
Unidad Profesional "Adolfo López Mateos", Zacatenco
07738, México D.F., México

http://www.ipn.mx
http://www.cic.ipn.mx

**Printing: 500**
**Impresiones: 500**

**Printed in Mexico**
**Impreso en México**

# Preface

Artificial Intelligence is a branch of computer science aimed at providing the computer elements of human-like behavior such as ability to think, learn by example, doubt, act, see, and speak. Since its beginning artificial intelligence research has been influenced and inspired by nature—in the first place, by the way human being accomplishes such tasks. Recently, the repertoire of artificial intelligence methods was enriched by other naturally inspired optimization techniques, such as genetic algorithms, swarm intelligence, or ant colony optimization. In addition to creating human-like-behaving machines, modern artificial intelligence provides a very powerful platform for solving a wide range of super-complex optimization problems.

This volume presents original research papers on the internal art and craft of artificial intelligence research: its theoretical foundations, specific techniques, and research methodologies. It is structured into eight thematic fields representative of the main current areas of interest within the AI community: Knowledge Representation and Logic; Constraint Satisfaction; Multiagent Systems and Distributed AI; Computer Vision and Pattern Recognition; Machine Learning and Neural Networks; Evolutionary Computation and Genetic Algorithms; Natural Language Processing; Modeling and Intelligent Control. The next volume of this journal presents original papers devoted to application of artificial intelligence techniques to practical real-life problems, from economy and education to creating of physical intelligent robots.

Total of 59 full papers by 145 authors from 20 different countries were submitted for evaluation, see Tables 1 and 2. Each submission was reviewed by three independent members of the Editorial Board of the volume. This volume contains revised versions of 26 papers, by 64 authors, selected for publication after thorough evaluation. Thus the acceptance rate was 44%. In Table 1, the number of papers by country was calculated by the shares of all authors of the paper: e.g., if a paper has three authors: two from Mexico and one from USA, then we incremented the counter for Mexico by 0.66 (two authors of three) and the counter for USA by 0.33. Table 2 presents the statistics of papers by topics according to the topics indicated by the authors; note that a paper can be assigned more than one topic.

**Table 1. Statistics of authors and papers by country.**

| Country | Submitted | | Accepted | | Country | Submitted | | Accepted | |
|---|---|---|---|---|---|---|---|---|---|
| | Auth | Pap | Auth | Pap | | Auth | Pap | Auth | Pap |
| Algeria | 2 | 0.6 | 2 | 0.6 | Lebanon | 2 | 1 | 2 | 1 |
| Argentina | 1 | 1 | – | – | Mexico | 43 | 16.8 | 30 | 12 |
| Brazil | 11 | 4 | – | – | Netherlands | 1 | 0.2 | – | – |
| China | 24 | 10 | 5 | 2 | Poland | 4 | 1 | – | – |
| France | 8 | 3.3 | 1 | 0.3 | Russia | 2 | 2 | – | – |
| Germany | 1 | 1 | – | – | Spain | 8 | 1 | 4 | 1 |
| India | 4 | 2 | 2 | 1 | Sweden | 4 | 1 | – | – |
| Italy | 2 | 1 | 2 | 1 | Tunisia | 2 | 1 | – | – |
| Japan | 5 | 2 | 5 | 2 | UK | 2 | 1 | 2 | 1 |
| Korea, South | 9 | 4 | 2 | 1 | USA | 10 | 4 | 7 | 3 |
| | | | | | total: | 145 | 59 | 64 | 26 |

**Table 2. Statistics of submitted and accepted papers by topic.**

| Topic | Submitted | Accepted |
|---|---|---|
| Expert Systems / KBS | 2 | – |
| Multiagent systems and Distributed AI | 10 | 6 |
| Knowledge Management | 3 | 1 |
| Intelligent Interfaces: Multimedia, Virtual Reality | 1 | – |
| Natural Language Processing / Understanding | 4 | 3 |
| Computer Vision | 4 | 2 |
| Neural Networks | 11 | 4 |
| Genetic Algorithms | 9 | 6 |
| Fuzzy logic | 4 | 1 |
| Machine Learning | 18 | 7 |
| Intelligent Tutoring Systems | 1 | – |
| Data Mining | 4 | – |
| Knowledge Acquisition | 3 | – |
| Knowledge Representation | 10 | 4 |
| Knowledge Verification, Sharing and Reuse | 3 | 1 |
| Ontologies | 4 | – |
| Qualitative Reasoning | 1 | 1 |
| Constraint Programming | 6 | 2 |
| Common Sense Reasoning | 1 | – |
| Case-Based Reasoning | 1 | – |
| Nonmonotonic Reasoning | 2 | 1 |
| Spatial and Temporal Reasoning | 1 | 1 |
| Robotics | 2 | 2 |
| Planning and Scheduling | 5 | 3 |
| Navigation | 5 | 2 |
| Hybrid Intelligent Systems | 4 | 1 |
| Logic Programming | 2 | 1 |
| Intelligent Organizations | 3 | 1 |
| Uncertainty / Probabilistic Reasoning | 1 | – |
| Bioinformatics | 3 | – |
| Philosophical and Methodological Issues of AI | 5 | 1 |
| Other | 10 | 4 |

The academic and editorial effort resulting in this volume was carried out in collaboration with, and was supported by, the Mexican Society for Artificial Intelligence (SMIA). We cordially thank all people involved in its preparation. In the first place these are the authors of the papers constituting it: it is the excellence of their research work that gives sense to the work of all other people involved. We thank the members of the Editorial Board of the volume and additional referees. We express our gratitude to Álvaro de Albornoz, Ángel Kuri, Hugo Terashima-Marín, Francisco J. Cantú-Ortiz, Leticia Rodríguez, Fernando J. Jaimes, Rogelio Soto-Rodríguez, Hiram Calvo, Manuel Vilares, and Sulema Torres for their significant contribution at various stages of preparation of the volume. The submission, reviewing, and selection process was supported for free by the EasyChair system, www.EasyChair.org.

Alexander Gelbukh
Raúl Monroy

November 2005

# Table of Contents
Índice

Page/Pág.

## Knowledge Representation and Logic

XML based Extended Super-function Schema in Knowledge Representation ..... 3
*Qiong Liu, Xin Lu, Fuji Ren and Shingo Kuroiwa*

The Description Logic of Tasks ..... 13
*Zhang Hui and Li SiKun*

A Logic Programming Formalization for Circumscription..... 23
*Masahiko Arai*

## Constraint Satisfaction

Genetic Algorithms for Dynamic Variable Ordering in Constraint Satisfaction Problems ..... 35
*Hugo Terashima-Marín, René de la Calleja-Manzanedo and Manuel Valenzuela-Rendón*

Two Hybrid Tabu Scatter Search Meta-heuristics for Solving MAX-SAT Problems..... 45
*Dalila Boughaci, Drias Habiba and Benhamou Belaid*

Complete Instantiation Strategy for Approximate Solution of TCSP55 ..... 55
*Priti Chandra and Arun K. Pujari*

Graphplan Based Conformant Planning with Limited Quantification ..... 65
*Alan Carlin, James G. Schmolze and Tamara Babaian*

## Multiagent Systems and Distributed AI

A Distributed Multiagent Workflow System ..... 79
*Cesar Marin and Ramon Brena*

Cooperation in multirobotics enviroments ..... 89
*Félix Orlando Martínez Ríos and Carlos Rodríguez Lucatero*

Economics of Cooperation : Social Foraging
in Distributed Systems ..... 99
*Ashish Umre and Ian Wakeman*

## Computer Vision and Pattern Recognition

Some Experiments on Corner Tracking for Robotic Tasks .....111
*Victor Ayala-Ramirez, Cruz A. Longoria-Mendez and Raul E. Sanchez-Yanez*

Pattern Decomposition and Associative Processing Applied
to Object Identification.....121
*Benjamín Cruz, Humberto Sossa and Ricardo Barrón*

Object Classification Based on Associative Memories and
Midpoint Operator.....131
*Roberto A. Vázquez, Humberto Sossa and Ricardo Barrón*

Associative Processing Applied to Word Reconstruction in the
Presence of Letter Scrambling .....141
*Humberto Sossa, Ricardo Barrón and Benjamín Torres*

## Machine Learning and Neural Networks

Hybrid strategies and meta-learning: an inquiry into the
epistemology of artificial learning .....153
*Ciro Castiello and Anna Maria Fanelli*

Comparison of Neuro-Fuzzy Systems with a Defuzzification-Based
Algorithm for Learning Fuzzy Rules .....163
*Jean Saade and Adel Fakih*

Evolutionary Computation and Genetic Algorithms Intelligent
Genetic Algorithm: A Toy Model Application .....175
*Jaime Mora Vargas, Neil Hernández Gress and Miguel González Mendoza*

Improved Ant Colony System using Subpath Information for the Taveling Salesman Problem ..... 185
*Minyoung Yun and Inkyeom Kim*

Building Block Filtering Genetic Algorithm ..... 195
*Jun Lu, Boqin Feng and Bo Li*

Evolutionary Training of SVM for Classification Problems with Self-Adaptive Parameters ..... 207
*Angel Kuri-Morales and Iván Mejía-Guevara*

## Natural Language Processing

Language Portable Detection for Spanish Named Entities ..... 219
*Zornitsa Kozareva, Oscar Ferrández, Andrés Montoyo and Rafael Muñoz*

Boosting Applied to Spanish Word Sense Disambiguation ..... 229
*Rocio Guillen*

A Cognitive-Based Approach to Adaptive Intelligent Multiagent Applications ..... 239
*Charles Hannon*

## Modeling and Intelligent Control

Orthogonal-Back Propagation Hybrid Learning Algorithm for Interval Type-2 Non-Singleton Type-2 Fuzzy Logic Systems ..... 251
*Gerardo M. Mendez and Luis A. Leduc*

Characterization and Interpretation of Classes Based on Fuzzy Rules in ill-Structured Domains ..... 261
*Fernando Vázquez and Juan Luis Díaz de León*

Differential Evolution Algorithms to Solve Optimal Control Problems Efficiently ..... 271
*Irineo L. Lopez-Cruz and Abraham Rojano-Aguilar*

# Knowledge Representation and Logic

# XML based Extended Super-function Schema in Knowledge Representation

Qiong Liu, Xin Lu, Fuji Ren and Shingo Kuroiwa

The University of Tokushima,
2-1 Minami Josanjima, Tokushima, Japan 770-8506
{liuqiong,luxin,ren,kuroiwa}@is.tokushima-u.ac.jp

**Abstract.** In recent years, the usual knowledge representation (KR) problem in artificial intelligence is how to automatically represent and transform different kinds of knowledge using one kind of schema. Especially this problem focuses on representing formal knowledge in natural language for human understanding. For this purpose, this paper proposes an extended super-function (ESF) schema to build a novel KR system. This system can translate the data of stock market or other fields into the corresponding natural language expression automatically. Moreover, this system benefits from XML techniques which formalize and construct all information using the common Web rules to realize the ESF schema.

## 1 Introduction

In artificial intelligence (AI), knowledge representation (KR) includes two basic kinds of knowledge objects (formal objects and natural objects) in its fundamental conception. Formal objects like mathematical entities can be captured exactly and precisely by machine because of their formality. Natural objects like natural language entities can be understood easily and commonly by human being through their flexibility. Then KR provides the representation function to deal with the correspondences between the formal objects and the natural objects, acting as surrogates in the real world as well as in the machine space. Given the relationship with human and machine is made closer, the last role of KR will become more significant and necessary.

In recent years, the KR technique has shown its superiority in knowledge collection and organization. The natural objects in knowledge base have been organized in highly structured form to satisfy the requirements that people wish to understand and master various kinds of knowledge easily by natural objects. The KR system based on natural objects is most sophisticated, and its construction is depended on some kinds of logic. Different formal methods (such as predicate logic, fuzzy logic, semantic networks, frames and related techniques) have been developed to represent natural objects. They also have been used by expert systems frequently in decision making and reasoning. The KR system [1–5] based on natural objects has been implemented in almost every aspect such as weather forecast, letters response, network analysis, disease diagnosis and so on. All of these systems have been recognized as the considerable enhancement of KR technique for natural objects management. Because natural objects like human natural language are complex, irregular, and diverse, the previous KR systems based on natural objects just act as interfaces to knowledge base, which perform formal tasks

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 3-12*

separated from nature objects processing. Nevertheless the computational characteristic of representation and inference in natural objects can improve the efficiencies of processing all tasks in the KR system. Therefore, the extended super-function (ESF) schema is proposed to build a novel KR system which processes vast amounts of knowledge systematically like machine and logically, deeply like human being. It is capable to incorporate both the formal objects and the natural objects. We think the ESF schema is a new direction in KR technique.

The paper is structured as follow. Section 2 describes the grammar of ESF schema for KR. In Section 3, the ESF based KR system is realized as example, where the ESF schema is utilized to produce technical report of stock market from data for non-expert user. Finally, Section 4 presents a discussion and conclusion of this paper.

## 2 Extended Super-function Schema

In ESF schema definition, symbol set includes formal objects and natural objects. Natural language is a symbol set, mathematic expression is a symbol set, music is a symbol set, and so on. The ESF schema is applied to the translation from one symbol set to another more variously than from one kind of natural language to another as SF.

An ESF is a symbol set that denotes the correspondence between source symbol patterns and target symbol patterns. The definition of symbol pattern is most necessary for ESF, and it will be described firstly. Then we give ESF a definition.

**Definition 1**: A team of a token $n$, some attributes $A$ and corresponding values $v$ can form a symbol pattern $p_n ::= p[n,\ A : v]$.

**Definition 2**: A symbol pattern $p$ can be sorted into source pattern $ps$ and target pattern $pt$ according to its origin. A set of source pattern $ps$ and a set of target pattern $pt$ can aggregate a source pattern set $Ps$ and a target pattern set $Pt$ respectively.

**Definition 3**: A function from $Ps_1 \times \cdots \times Ps_{n-1}$ to $Ps_n$ is a subset $fs$ of the Cartesian product $Ps_1 \times \cdots \times Ps_n$, such that for each pair $(ps_1, \ldots, ps_{n-1})$ in $Ps_1 \times \cdots \times Ps_{n-1}$, there is a unique $ps_n$ in $Ps_n$ such that the ordered pair $(ps_1, \ldots, ps_n)$ is in $fs$. The source pattern $ps_n$ can be described as $ps_n ::= ps[n,\ A : fs]$

**Definition 4**: A function from $Pt_1 \times \cdots \times Pt_{n-1}$ to $Pt_n$ is a subset $ft$ of the Cartesian product $Pt_1 \times \cdots \times Pt_n$, such that for each pair $(pt_1, \ldots, pt_{n-1})$ in $Pt_1 \times \cdots \times Pt_{n-1}$, there is a unique $pt_n$ in $Pt_n$ such that the ordered pair $(pt_1, \ldots, pt_n)$ is in $ft$. The source pattern $pt_n$ can be described as $pt_n ::= pt[n,\ A : ft]$

**Definition 5**: A function from $Ps$ to $Pt$ is a subset $r$ of the Cartesian product $Ps \times \cdots \times Pt$, such that for each $ps$ in $Ps$, there is a unique $pt$ in $Pt$ such that the ordered pair $(ps, pt)$ is in $r$.

**Definition 6**: A set of function $fs$ can aggregate a source function set $Fs$. A set of function $fs$ can aggregate a target function set $Ft$ in the same way. A set of function $r$ can be considered as relation set $R$. $R$ means the translations between source patterns and target patterns.

In ESF, the based element is atomic pattern. Any symbol pattern, whose value is obtained from function, can be defined as a complex pattern. Its value is composed of some atomic patterns or other complex patterns ordered in function structures. Therefore, these symbol patterns are not ordered in one layer, they are ordered like net by functions. The grammar of ESF can be set as a five-tuple.

$$< Ps,\ Pt,\ Fs,\ Ft,\ R >$$

## 3 KR System Construction using ESF Schema

We can master the ESF based KR system from three layers (i.e., abstracting layer, describing layer and implementing layer). The three layers is used for understanding, detailing and coding the KR system respectively. In this section we specify KR system in these three layers for understanding.

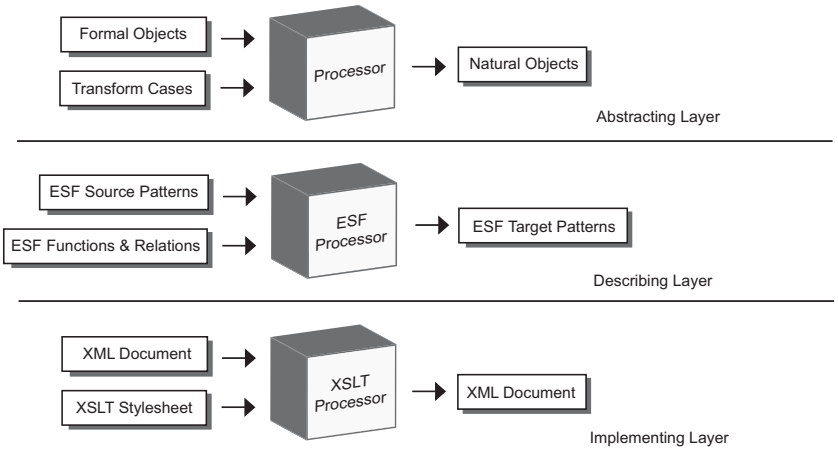



**Fig. 1.** Fundamental structure of the KR system based on ESF schema

As shown in the top layer of Figure 1, the processor receives the formal objects with the transform cases, and generates the natural objects. This is a pipeline mechanism. Here, the transform cases are regarded as principles for guiding processor what to do and how to do. They are the crucial parts of KR system.

### 3.1 Transform Case Extraction in Abstracting Layer

There are hundreds s of indicators in use today. Every technical indicator can be regarded as one transform case in the abstracting layer of our KR system. The technical indicator data and its perspective can be considered as the formal objects and the natural objects of transform case respectively. In this paper, Moving average convergence

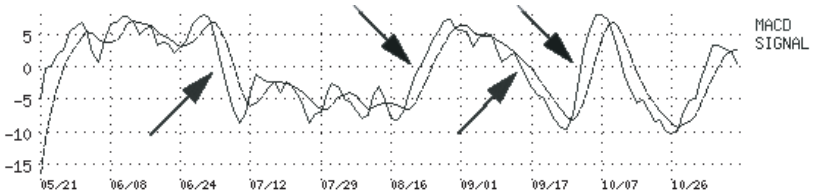



**Fig. 2.** Moving Average Convergence Divergence i)Upper-oriented arrows marks bearish centerline crossover. ii) Lower-oriented arrows marks bullish centerline crossover.

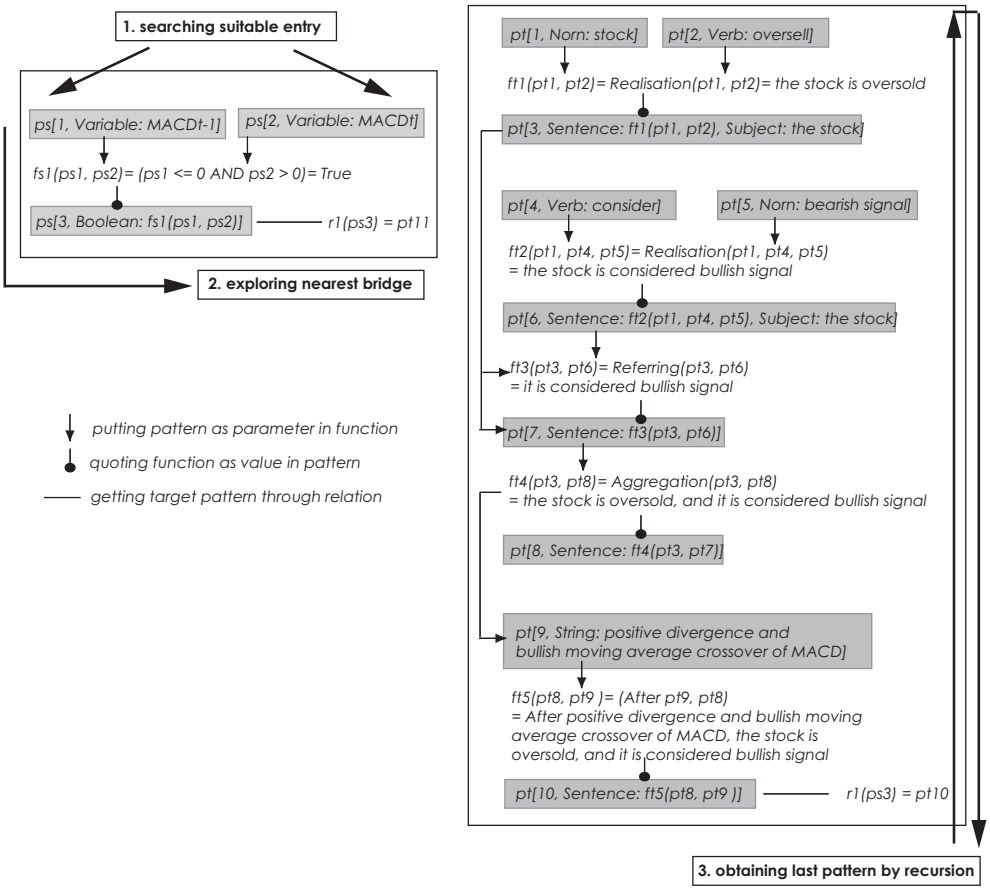



**Fig. 3.** ESF structures of MACD indicator (source patterns and source functions (left), target patterns and target functions (right))

divergence (MACD) is extracted as a transform cases from our KR system for clear specification.

A bullish centerline crossover occurs when MACD moves above the zero line and into positive territory. This is a clear indication that momentum has changed from negative to positive or from bearish to bullish. After a positive divergence and bullish moving average crossover, the bullish centerline crossover can act as a confirmation signal. Conversely, after a negative divergence and bearish moving average crossover, the bearish centerline crossover can act as a confirmation signal. (The MACD chart is illustrated in Figure 2). Then we can get a transform description of MACD:

```
IF MACD(Day)≥0 AND MACD(Day-1)<0 THEN "after a positive divergence
and bullish moving average crossover of MACD, the bullish centerline
crossover can act as a confirmation signal"
IF MACD(Day)≤0 AND MACD(Day-1)>0 THEN "after a negative divergence
and bearish moving average crossover of MACD, the bearish centerline
crossover can act as a confirmation signal"
```

### 3.2 ESF Structure Design in Describing Layer

From abstracting layer to describing layer, we utilize the ESF schema to design the ESF structure of our KR system detailedly. All non-linguistic inputs are defined as ESF source pattern; the transform cases are generalized as ESF functions and relations for reuse. The ESF processor can accelerate the map between the source patterns and the target patterns through the functions and relations as illustrated in the middle layer of Figure 1.

Here a transform case (i.e., MACD indicator mentioned above) has been generalized as corresponding functions and relations by ESF schema in Figure 3 as an example. In this example every block denotes a pattern, such as atomic or complex one. Every relation can be considered as a translation bridge between numerable source patterns and single target pattern. Every function acts as telephone line which is connecting every user – pattern. We need to emphasize that the process from pattern to function means putting pattern into function as parameter. On the other hand, the process from function to pattern is quoting function as value in pattern. These two processes will be coded in computer language for the actual experiment in following section. The ESF process consists of three steps. They are marked in Figure 3.

1. **Searching suitable entry:** We put some source patterns, which will be translated into natural language, into processor. Then the processor searches source functions' parameters as the suitable entries for these source patterns. Here the "suitable" denotes the pattern's domain is consistent with a parameter's domain or contained by it. In this example, if the source patterns include other technical indicator information except the $MACD$, it is difficult for processor to find the suitable entries.

2. **Exploring nearest bridge:** After searching suitable entry, the processor explores downwards, passes numerable functions and finds a nearest relation. Then the processor can obtain a corresponding target pattern through the relation bridge. There is a necessary condition in this process. It is all of the numerable functions' parameters (from suitable entry to nearest relation) should be filled by source patterns. In this example, if there is only the $MACD(Day)$ pattern $ps_1$ and not the $MACD(Day-1)$

pattern $ps_2$, the processor cannot find the relation $r_1$.

3. **Obtaining last pattern by recursion:** If the processor gets the corresponding target pattern, it will use the correlative target functions structured recursively to get the corresponding target pattern's value.

ESF based KR system is a new KR system that has the characters between template-based and standard KR. This is not only because the ESF based KR system combines standard KR with templates, but also because it tends to use syntactically structured templates (here the "template" in ESF schema is a function. It has more changes than the common template), and allows the gaps as parameters in them to be filled recursively (i.e., by filling a gap, a new gap may result). The ESF based KR system can use grammars to aid linguistic realization. For example, in Figure 5, it includes lexical items (e.g., referring expression function, aggregation function and so on) which always exist in standard KR. Therefore, it is difficult to give a definition of "template based" for our KR system. We think the word – "function based" is more suited than "template based".

### 3.3 ESF Structure Realization in Implementing Layer

It is necessary to utilize a ready-made and convenient technique to realize ESF schema in KR systems. We regard the extensible markup language (XML) is the best choice to define the ESF pattern. We also select extensible stylesheet language transformation (XSLT) to describe the ESF function and relation. This section helps the reader master and apply these ideas to KR problems. We utilize the XML and XSL techniques to realize ESF schema for building a KR system. As shown in the bottom layer of Figure 1 , in a KR system the ESF patterns in XML is fed into the XSLT processor as one input, and the functions and relations in XSL is provided as a second input. The output is then sent directly to user as a stream of HTML, XML or other formats. The ESF functions and relations in XSL generate the transformation instructions about ESF patterns, and the patterns in XML provide raw data. It is the implementing layer of the KR system based on ESF schema.

**Pattern construction using XML:**

Code 1 – ESF source patterns

```
<stock symbol="TOPIX" name="Tokyo Stock Exchange Prices Index">
...
<date="2004-08-20" open="1105.08" close="1109.84" volume="1072250000"
MACD="-1.8351"/>
<date="2004-08-23" open="1115.93" close="1114.24" volume="1047230000"
MACD="0.1509"/>
<date="2004-08-24" open="1116.74" close="1116.60" volume="1065260000"
MACD="1.5825"/>
...
</stock>
```

We begin with an XML document that represents a portion of quotations about MACD in stock market, which is shown above. The XML elements include their at-

tributes with their respective values within the element's start tag. Because the ESF pattern's attribute has the same name-value form, it can map the corresponding XML element's attribute with value. Therefore, all XML elements can be considered as the ESF patterns.

**Function and relation generation using XSLT:**

Here the centric problems are how to build the XSLT stylesheet for realizing ESF functions and relations, then how to process XML documents including ESF patterns by the XSLT stylesheet. In this paper we utilize XSLT templates to do this work. In Code 2, 3 and 4, we write some templates for realizing the ESF functions and relations of the MACD transform case (shown in Figure 3), and process the XML document mentioned above by any XSLT processor.

The XSLT templates are always written in one XSLT file together. We separate it into three pieces in Code 2, 3 and 4 as their corresponding ESF roles for understanding easily. Code 2, 3 and 4 respectively show source functions set, relations set, and target functions set. The process of ESF is from "searching suitable entry in source functions set, to "exploring nearest bridge" in relation set, then to "obtaining last pattern by recursion" in target function set. For this example we describe detailed technological process as following:

**1** Source patterns extraction in Code 2 – (Searching suitable entry)

Code 2 – ESF source function

```
<xsl:template name="fs1" match="MACD">
<xsl:variable name="ps1"/>
<xsl:value-of select="//ps1@value"/>
</xsl:variable>
<xsl:variable name="ps2">
<xsl:value-of select="//ps2@value"/>
</xsl:variable>
<xsl:choose>
<xsl:when test="$ps1 &lt;= 0 and $ps2 &gt; 0">
<xsl:value-of select="1"/>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="0"/>
</xsl:otherwise>
</xsl:choose>
</xsl:template>
```

Put two patterns $ps_1,\ ps_2$ into source function $fs_1\ (ps_1,\ ps_2) = (ps_1 \leq 0, ps_2 > 0)$, and quote the function value as new pattern $ps_3$. The obtained patterns $ps_3$ are regarded as source patterns.

**2** Transformation from source patterns to target patterns in Code 3 – (Exploring nearest bridge)

Transform source pattern $ps_3$ into target pattern $pt_{10}$ through relation $r_1(ps_3) = pt_{10}$. In the transformation, the target patterns $pt_{10}$, is needed.

Code 3 – ESF relation

```
<xsl:template name="r1">
<xsl:variable name="macd">
<xsl:value-of select="//ps3@value"/>
</xsl:variable>
<xsl:choose>
<xsl:when test="$macd = 1">
</xsl:call-template name="fs5">
</xsl:when>
</xsl:choose>
</xsl:template>
```

**3** Target patterns extraction in Code 4 – (Obtaining last pattern by recursion)

Code 4 – ESF target functions

```
<xsl:template name="ft3">
<xsl:choose>
<xsl:when test="//pt3@subject = //pt6@subject">
<xsl:call-template name="search-and-replace"/>
<xsl:with-param name=" input " select="//pt6@sentence"/>
<xsl:with-param name="search-string" select="//pt6@subject"/>
<xsl:with-param name="replace-string" select="it"/>
</xsl:call-template >
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="//pt6@sentence"/>
</xsl:otherwise>
</xsl:choose>
</xsl:template>

<xsl:template name="ft4">
<xsl:value-of select="//pt3@sentence"/>
<xsl:text>, and </xsl:text>
<xsl:value-of select="//pt7@sentence"/>
</xsl:template>

<xsl:template name="ft5">
<xsl:text>After </xsl:text>
<xsl:value-of select="//pt9@sentence"/>
<xsl:text>, </xsl:text>
<xsl:value-of select="//pt8@sentence"/>
</xsl:template>
```

Put pattern $pt_3$, $pt_6$ into target function $ft_3\ (pt_3,\ pt_6) = Referring(pt_3,\ pt_6)$, and quote the function value as new pattern $pt_7$.

Then put pattern $pt_3$, $pt_7$ into target function
$ft_4\ (pt_3,\ pt_7) = Aggregation(pt_3,\ pt_7)$, and quote the function value as new pattern $pt_8$.

Then put pattern $pt_8$, $pt_9$ into target function
$ft_5\ (pt_8,\ pt_9) = (After\ pt_9,\ pt_8)$, and quote the function value as new pattern $pt_{10}$. The obtained patterns $pt_{10}$ is regarded as target patterns.

These XSLT codes mentioned above are just a portion of all. If the XML document constructed as Code 1, then all of XSLT codes are implemented, the result text can be obtained as shown in follows:

Result

```
"In Tokyo Stock Exchange market, the Tokyo Stock Exchange Prices Index
(TOPIX) is analyzed now. ...
It opened at 1105.08, and closed at 1109.84 on 2004-08-20. Its total
turnover was 1072250000. It opened at 1115.93, and closed at 1114.24 on
2004-08-23. Its total turnover was 1047230000. After positive divergence
and bullish moving average crossover of MACD, the stock is oversold,
and it is considered bullish signal. It opened at 1116.74, and closed at
1116.60 on 2004-08-24. Its total turnover was 1065260000. ..."
```

## 4 Discussion and Conclusion

Our hypotheses are that texts which contain technical indicators as described above will help non-experts to retain more information and perform better than charts, and that non-experts will rate these texts as more interesting and pleasant to read. For this point, the evaluation experiment is carried out in which learning outcomes of texts will compare with charts'. The MACD indicator is chosen for the evaluation. For each indicator two evaluation suites are prepared. The only difference between them is that one uses the texts of technical indicators and the other uses the charts.

The tested are 40 students who do not have expert knowledge of stock market. They are separated into two equally sized groups. Group A read the charts where the technical indicators are marked and group B read the analysis texts which our KR system generates. After reading, two groups will finish the evaluation suite including "comprehension", "accuracy", "time", "interest", "remembrance", and "usefulness" items. The evaluation results are shown in Figure. Obviously, the "comprehension", "interest", "remembrance" items' scores of group B are much higher than group A. Because the tested think not only charts but also texts are necessary for the technical indicators to be described, the "usefulness" item's score between group A and group B is about same. Otherwise, the "accuracy", "time" items' scores of group B approach group A. The reason is that the ESF schema has linguistic completeness and the computer's speed is much faster than before. The ESF based KR system can generate sufficiently quality analysis texts for non-experts.

In the traditional view, KR can be separated into two kinds: *Template based* KR and *standard* KR. Template based KR system maps its non-linguistic input directly (i.e., without intermediate representations) to the linguistic surface structure. Crucially, this linguistic structure may contain gaps. Well-formed output can be obtained when the gaps of linguistic structure are filled until the linguistic structure does not contain gaps. By contrast, standard KR systems use a less direct mapping between input and surface

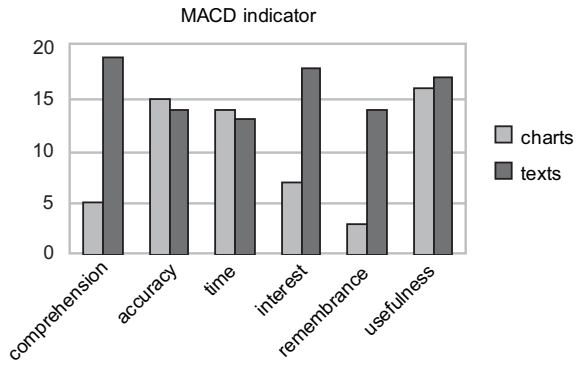



**Fig. 4.** evaluation experiment

form. Such systems could begin with inputting semantic representation, subjecting it to a number of consecutive transformations until a surface structure results.

Within this paper, the ESF based KR system can do the accurate and convenient transformation between all kinds of knowledge objects. Because not only the ESF schema combines standard KR with templates, but also it tends to use syntactically structured "templates" – function (here the function has more changes than the common template), and allows the gaps as function's parameters to be filled recursively for realizing linguistic expression like standard KR system. Therefore, we name our KR system – *function based* KR system.

Furthermore, we utilize the XML and XSTL techniques to describe the ESF schema in this paper. Because the XML and XSTL have become the main techniques of information formalizing on Web, they supply the common rules for developers to format all information under one standard. Therefore, this schema using XML and XSTL techniques can be realized on Web for information's integration, distribution and transformation.

## References

[1] V. Mittal, J. Moore, G. Carenini and S.F. Roth, Describing Complex Charts in Natural Language: A Caption Generation System *Computational Linguistics* (1998, 24(3), pp. 431-467).

[2] M. Theune, E. Klabbers, J. Pijper, E. Krahmer and J. Odijk, From data to speech: a general approach, *Natural Language Engineering* (2001, 7(1), pp. 47-86).

[3] Susan W. McRoy, S. Channarukul and Syed S. Ali, An augmented template-based approach to text realization, *Natural Language Engineering* (2003, 9(4), pp. 381-420).

[4] M. White, Reining in CCG Chart Realization, in *Proceedings of the 3rd International Conference on Natural Language Generation* (2004).

[5] Mary E. Foster and M. White, Assessing the Impact of Adaptive Generation in the COMIC Multimodal Dialogue System. in *Proceedings of the Workshop on Knowledge and Reasoning in Practical Dialogue Systems* (2005).

# The Description Logic of Tasks[1]

Zhang Hui  Li Sikun

College of Computer Science,
National University of Defense Technology,
Changsha, China
qd_zhanghui@163.com

**Abstract.** The logic of tasks can be used in AI as an alternative logic of planning and action, its main appeal is that it is immune to the frame problem and the knowledge preconditions problem other plan logics confront. A drawback the present logic of tasks has is that it is nondecidable (semidecidable only). This paper extends the logic of tasks to enable the task description by adapting description structure into it. A formal system DTL, which is sound, complete and decidable, for agent abilities specification and accomplishablity of tasks judgment is proposed.

## 1 Introduction

The semantic used in the logic of tasks can claim to be a formalization of the resource philosophy associated with linear logic[4,5,6], if resources are understood as agents carrying out tasks. The formalism may also have a potential to be used in AI as an alternative logic of planning and action and its main appeal is that it is immune to the frame problem and the knowledge preconditions problem other plan logics confront [2,10]. A drawback the present logic of tasks has is that it is nondecidable (semidecidable only).

Motivated by the success of description logic[7.8.9], in this paper we present a decidable logic system, the description logic of tasks that enable task description by adapting the description structure into the logic of tasks. In our paper the task may have parameter e.g. we can use C($x$) denoting the task to clean $x$, $x$ can be either room or lawn (room and lawn are constant). The expression $\alpha\rightarrow\beta$ in our paper means that the accomplishment of task $\alpha$ is the condition to accomplish the task $\beta$, such as F(rake)$\rightarrow$C(lawn) and F(mop)$\rightarrow$C(room) express that the cleaner can clean the room if be given a mop and can clean the lawn if be given a rake respectively.

A characteristic feature of description languages is their ability to represent other kinds of relationships, expressed by role, beyond IS-A relationships. The role has what is called a "value restriction," denoted by the label $\forall$R. which expresses a limitation on the range of types of objects that can fill that role. In this paper we use role to express the relation between objects. For example R(room,mop) means that

[1] Supported by the National Grand Fundamental Research 973 Program (2002CB312105) and 863 program (2004AA115130) of China.

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 13-22*

there is certain relation between room and mop (a mop is the necessity tool to clean a room). The role has what is called "value restriction" denoted by the label ∀R. too, which also expresses the limitation on the range of object that can fill the role. For example, the expression ∀R(room,y).F(y)→C(room) means that if be given all necessity tools for cleaning a room the cleaner can accomplish the task of cleaning it.

In addition, we use predict to express the limitation on the range of value of the parameter of tasks. For example, if we used predict P to express whether the object is in charged of by the cleaner or not, then ∀P(x).(∀R(x,y).F(y)→C(x)) expresses that for every object *x* that the cleaner in charge of, if be given all necessity tools the cleaner can accomplish the task of cleaning *x*.

In remain of the paper we will give out the syntax and the semantic of the description logic of tasks.

# 2 Syntax and Semantic of the Description Logic of Tasks

### 2.1. Syntax

We fix a set of expressions that we call atom task names $\{A,A_1,A_2\ldots\}$, with each of which is associated a natural number called its arity, a set of predict name$\{P,P_1, P_{2,}\ldots\}$, with each of which also is associated a natural number called its arity, a set of role names $\{ R,R_1,R_2,\ldots\}$.

We also fix an infinite set $C=\{c_0, c_1, \ldots\}$ of constants and an infinite set $X=\{x_0, x_1, x_2, \ldots\}$ of variables. We refer to elements of $C \cup X$ as terms.

**Definition 2.1 (**domain knowledge axioms**)** Domain knowledge axioms are defined as follows:

1. Let *a, b* be two constants and R a role name, then R(a,b) is a domain knowledge axiom;
2. Let $c_1, c_2, \ldots, c_n$ be constants and P an n-ary predict then $P(c_1,c_2,\ldots,c_n)$ is a domain knowledge axiom.

**Definition2.2** (task formula) Task formulas are elements of the smallest class of expressions such that:

1. If A is a n-ary atom task name, $t_1,t_2,\ldots,t_n$ are terms, then $A(t_1,t_2,\ldots,t_n)$ is task formula, be called an atomic task;
2. if P is a n-ary predict, α is a formula, $t_1,t_2,\ldots,t_n$ are terms, then $\forall P(t_1,t_2,\ldots,t_n).\alpha$ and $\exists P(t_1,t_2,\ldots,t_n).\ \alpha$ are both task formulas;
3. If α is a formula, R is an role, *t* is a term and *y* is a variable, then $\forall R(t,y).\alpha$ and $\exists R(t,y).\alpha$ are both task formulas;
4. if α and β are task formulas, then so are $\alpha\wedge\beta,\alpha\vee\beta$ and $\alpha\rightarrow\beta$;
5. if α and β are task formulas, then so is $\alpha\sqcap\beta$;
6. if α is a formula, *x* is a variable, then $\Pi x\alpha$ is a task formula.

$\sqcap$ and $\Pi$ are called additive operators, or additives. The additive complexity of a formula is the number of occurrences of additives in that formula. The task formula whose complexity is zero is called primitive task formula and task formula that does not contain free variable is call to be closed.

Except the sets mentioned above, the description logic of tasks also have two other sets, the domain knowledge and the capability specification, defined as follows:

**Definition 2.3** (domain knowledge and capability specification) The Domain knowledge is a finite set of domain knowledge axioms. The capability specification is a finite set of primitive task formulas.

## 2.2 Semantic

Let $t_1,t_2,\ldots,t_n$ be terms, an assignment of $(t_1,t_2,\ldots,t_n)$ is a n-tuple $(c_1/t_1,c_2/t_2,\ldots,c_n/t_n)$ such that $c_i \in C$ and if $t_i$ is constant then $c_i=t_i$ for all i($1\le i\le n$). Let $\alpha$ be a primitive task formula, $\alpha(t_1/c_1,t_2/c_2,\ldots, t_n/c_n)$ is the result of replacing all free occurrences of $t_i$ in $\alpha$ by $c_i$ respectively (i=1,2,…,n) if $t_i$ is a variable.

We consider interpretation I that consist of a non-empty set $\Delta^I$ (the domain of the interpretation) and an interpretation function $\cdot^I$, which assigns to every predict P a set $P^I \subseteq (\Delta^I)^n$, to every role R a binary relation $R^I \subseteq \Delta^I \times \Delta^I$, to every closed atomic task an element of $\{0,1\}$. To give out the value of all closed primitive task formulas, $\cdot^I$ is extended as follows:

1. $(\neg\alpha)^I=\begin{cases}0 & \text{if } \alpha^I=1\\ 1 & \text{else}\end{cases}$;

2. $\forall P(t_1,t_2,\ldots,t_n).\alpha=$

$\begin{cases}1 & \text{if } (\alpha(t_1/c_1,t_2/c_2,\ldots,t_n/c_n))^I=1 \text{ for every assignment } (t_1/c_1,t_2/c_2,\ldots,t_n/c_n) \text{ such that } (c_1,c_2,\ldots,c_n)\in P^I\\ 0 & \text{else}\end{cases}$;

3. $\exists P(t_1,t_2,\ldots,t_n).\alpha=$

$\begin{cases}1 & \text{if there is an assignment } (t_1/c_1,t_2/c_2,\ldots,t_n/c_n) \text{ such that } (c_1,c_2,\ldots,c_n)\in P^I \text{ and } (\alpha(t_1/c_1,t_2/c_2,\ldots,t_n/c_n))^I=1\\ 0 & \text{else}\end{cases}$;

4. $(\forall R(a,y).\alpha)^I=\begin{cases}1 & \text{if } (\alpha(y/b))^I=1 \text{ for every } b \text{ such that } (a,b)\in R^I\\ 0 & \text{else}\end{cases}$;

5. $(\exists R(a,y).\alpha)^I=\begin{cases}1 & \text{if there exist a constant } b \text{ such that } (a,b)\in R^I \text{ and } (\alpha(y/b))^I=1\\ 0 & \text{else}\end{cases}$;

6. $(\alpha\wedge\beta)^I=\begin{cases}1 & \text{if } \alpha^I=1 \text{ and } \beta^I=1\\ 0 & \text{else}\end{cases}$;

7. $(\alpha\vee\beta)^I=\begin{cases}1 & \text{if } \alpha^I=1 \text{ or } \beta^I=1\\ 0 & \text{else}\end{cases}$;

8. $(\alpha\rightarrow\beta)^I=\begin{cases}0 & \text{if } \alpha^I=1 \text{ and } \beta^I=0\\ 1 & \text{else}\end{cases}$;

We say an interpretation I is coincident with the domain knowledge if it satisfies conditions follows:

1. For every n-tuple of constants $(c_1,c_2,\ldots,c_n)$, $(c_1,c_2,\ldots,c_n)\in P^I$ if and only if $P(c_1,c_2,\ldots,c_n)$ is in the domain knowledge;
2. for every 2-tuple of constants $(a,b)$, $(a,b)\in R^I$ if and only if $R(a,b)$ is in the domain knowledge.

Let $\Gamma$ be a set of primitive task formulas, $\{x_1,x_2,\ldots,x_n\}$ be the set of all free variable that occur in the task formulas in $\Gamma$. An interpretation I and an assignment $(x_1/c_1,x_2/c_2,\ldots,x_n/c_n)$ is said satisfy $\Gamma$ if $(\alpha(x_1/c_1,x_2/c_2,\ldots,x_n/c_n))^I=1$ for every formula $\alpha$ in $\Gamma$. A task formula set $\Gamma$ is said be not satisfiable if there is no interpretation I and assignment $(x_1/c_1,x_2/c_2,\ldots,x_n/c_n)$ satisfy it, else be satisfiable.

In remain of the paper we only consider the interpretation that is coincident with the domain knowledge and assumes that the ability specification is satisfiable.

**Definition 2.5**(accomplishability of primitive tasks) Let $\Gamma$ be the ability specification and $\alpha$ a primitive task. Let $\{x_1,x_2,\ldots,x_n\}$ be the set of all free variables that appear in the task formula in the set $\Gamma\cup\{\alpha\}$. We say that $\alpha$ is accomplishable if for every interpretation I and every assignment $(x_1/c_1,x_2/c_2,\ldots,x_n/c_n)$ of $(x_1,x_2,\ldots,x_n)$ that satisfy $\Gamma$, we have $(\alpha(x_1/c_1,x_2/c_2,\ldots,x_n/c_n))^I=1$.

Now we will give out the concept of accomplishable task. The concepts of strategy and realization used are same as those in [2].

Observe that development preserves the basic structure of the formula. I.e.

1. assume $\alpha_0=\forall P(t_1,t_2,\ldots,t_n).\beta$ (or $\alpha_0=\exists P(t_1,t_2,\ldots,t_n).\beta$), then for every realization $\mathcal{R}=<\alpha_0,\alpha_1,\alpha_2,\ldots,\alpha_m>$ of $\alpha_0$, $\alpha_i$ must has the form of $\forall P(t_1,t_2,\ldots,t_n).\beta_i$ (or $\exists P(t_1,t_2,\ldots,t_n).\beta_i$)) $(1\le i\le n)$. For every assignment $(t_1/c_1,\ t_2/c_2,\ldots,t_n/c_n)$ the sequence of $\beta_i(t_1/c_1,\ t_2/c_2,\ldots,t_n/c_n)$, denoted by $[P:\ t_1/c_1,\ t_2/c_2,\ldots,t_n/c_n]\ \mathcal{R}$, is an realization of $\beta(\ t_1/c_1,\ t_2/c_2,\ldots,t_n/c_n)$.
2. Assume $\alpha_0=\forall R(t,y).\beta$ (or $\alpha_0=\exists R(t,y).\beta$), then for every realization $\mathcal{R}=<\alpha_0,\alpha_1,\alpha_2,\ldots,\alpha_m>$ of $\alpha_0$, $\alpha_i$ must has the form of $\forall R(t,y).\beta_i$ (or $\exists R(t,y).\beta_i$) $(1\le i\le n)$ and for every constant $b$ the sequence of $\beta_i(y/b)$, denoted by $[R:y/b]\mathcal{R}$, is an realization of $\beta(y/b)$.
3. Assume $\alpha_0=\beta\wedge\gamma$(or $\alpha_0=\beta\vee\gamma$), then for every realization $\mathcal{R}=<\alpha_0,\alpha_1,\alpha_2,\ldots,\alpha_m>$ of $\alpha_0$, $\alpha_i$ can be expressed as $\beta_i\wedge\gamma_i$ (or $\beta_i\vee\gamma_i$) $(1\le i\le n)$ such that $<\beta_0,\beta_1,\beta_2,\ldots,\beta_m>$ and $<\gamma_0,\gamma_1,\gamma_2,\ldots,\gamma_m>$, denoted by $p(\mathcal{R})$ and $r(\mathcal{R})$, are realizations of $\beta$ and $\gamma$ respectively.
4. Assume $\alpha_0=\beta\to\gamma$, then for every realization $\mathcal{R}=<\alpha_0,\alpha_1,\alpha_2,\ldots,\alpha_m>$ of $\alpha_0$, $\alpha_i$ must has the form of $\beta_i\to\gamma_i$ $(1\le i\le n)$ . $<\beta_0,\beta_1,\beta_2,\ldots,\beta_m>$ and $<\gamma_0,\gamma_1,\gamma_2,\ldots,\gamma_m>$ denoted by $a(\mathcal{R})$ and $c(\mathcal{R})$, are realizations of $\beta$ and $\gamma$ respectively.

**Definition 2.6** We say that an realization $\mathcal{R}=<\alpha_0,\ \alpha_1,\ \alpha_2,\ldots,\ \alpha_m>$ of a task formula $\alpha_0$ is successful if one of the following conditions holds:

1. If $\alpha_0$ is an atomic task formula, or $\alpha_0=\neg\alpha$ and $\alpha$ is an atomic task formula(both imply m=0), and $\alpha_0$ is accomplishable;
2. $\alpha_0=\forall P(t_1,t_2,\ldots,t_n).\beta$ and $[P:\ t_1/c_1,\ t_2/c_2,\ldots,t_n/c_n]\ \mathcal{R}$ is successful for every assignment $(t_1/c_1,t_2/c_2,\ldots,t_n/c_n)$ such that $P(c_1,c_2,\ldots,c_n)$.
3. $\alpha_0=\exists P(t_1,t_2,\ldots,t_n).\beta$ and there is an assignment $(t_1/c_1,t_2/c_2,\ldots,t_n/c_n)$ such that $P(c_1,c_2,\ldots,c_n)$ and $[P:t_1/c_1,t_2/c_2,\ldots,t_n/c_n]\ \mathcal{R}$ is successful;

4. $\alpha_0=\forall R(t,y).\beta$ and $[R:y/b]\ \mathcal{R}$ is successful for every constant $b$ such that $R(t,b)$;
5. $\alpha_0=\exists R(t,y).\beta$ and there is a constant $b$ such that $R(t,b)$ and $[R:y/b]\ \mathcal{R}$ is successful ;
6. $\alpha_0=\beta\wedge\gamma$ and $p(\mathcal{R})$ and $r(\mathcal{R})$ both are successful;
7. $\alpha_0=\beta\vee\gamma$ and either $p(\mathcal{R})$ or $r(\mathcal{R})$ is successful;
8. $\alpha_0=\beta\rightarrow\gamma$ and $c(\mathcal{R})$ if successful if $a(\mathcal{R})$ is successful.
9. $\alpha_0$ is an additive formula and either $m=0$ or $m\geq 1$ and $<\alpha_1, \alpha_2,\ldots, \alpha_m>$ is successful.

**Definition 2.7** (accomplishability of tasks) Let $\alpha$ be a task formula. If there is an action strategy *f* such that every realization of $\alpha$ with *f* is successful then we say that $\alpha$ is accomplishable.

## 3 Accomplishablity judgment of primitive tasks

In this section the method for accomplishablity judgment of primitive tasks is presented. The work in this section is inspired and highly related to F. Baader and P. Hanschke's work for the consistent judgment of description logic formula set [9].

For a task formula $\alpha$, it is accomplishable if and only if that $\Gamma\cup\{\neg\alpha\}$ is not satisfiable. So here we need only give out the method for satisfiability judgment of finite set of primitive task formulas.

First, we assume that the task formula in the task formula set, denoted by $S_1$, does not has R(1) type free variable. A variable of task formula $\alpha$ is said to be R(1) type if *x* is a free variable and $\alpha$ has sub formula has the form $\forall R(x,y).\beta$ or $\exists R(x,y).\beta$. In fact if there is a R(1) type free variables *x* in $S_1$ , the number of constant *a* with a constant *b* such that R(a,b) is in domain knowledge is finite, assume $\{a_1,a_2,\ldots,a_n\}$ is the set of all such *a,* then $S_1$ is satisfiable if and only if at least one of the sets $S_1(x/a_i)$ is satisfiable.

Furthermore, we assume that every formula in $S_1$ is in negation normal form, i.e. $\neg$ occurs only immediately before the atom task name, in fact if a task formula in $S_1$ is not in negation normal form it can be transformed in to an equivalent one.

**Definition 3.1**(transformation rules) Let M be a finite set of finite primitive task formula sets. The following rule will replace one of elements S of M by another set (or several other sets) of primitive task formulas.

Rule 1: If $\alpha(c_1/t_1,c_2/t_2,\ldots,c_n/t_n)\in S$ for every assignment $(c_1/t_1,c_2/t_2,\ldots,c_n/t_n)$ such that $P(c_1,c_2,\ldots,c_n)$, $\forall P(t_1,t_2,\ldots,t_n).\alpha$ is a sub formula of one element of S and $\forall P(t_1,t_2,\ldots,t_n).\alpha$ is not in S, then replace S by $S'=S\cup\{\forall P(t_1,t_2,\ldots,t_n).\alpha\}$;

Rule 1′: if $P(c_1,c_2,\ldots,c_n)$, $\forall P(t_1,t_2,\ldots,t_n).\alpha\in S$ and $\alpha(c_1/t_1,c_2/t_2,\ldots,c_n/t_n)$ is not in S, then replace S by $S'=S\cup\{\alpha(c_1/t_1,c_2/t_2,\ldots,c_n/t_n)\}$;

Rule 2: if there is an assignment $(c_1/t_1,c_2/t_2,\ldots,c_n/t_n)$ of $(t_1,t_2,\ldots,t_n)$ such that $P(c_1,c_2,\ldots,c_n)$ and $\alpha(c_1/t_1,c_2/t_2,\ldots,c_n/t_n)\in S$, $\exists P(t_1,t_2,\ldots,t_n).\alpha$ is a sub formula of one element of S but $\exists P(t_1,t_2,\ldots,t_n).\alpha$ is not in S, then replace S by $S'=S\cup\{\exists P(t_1,t_2,\ldots,t_n).\alpha\}$

Rule 2′: if $\exists P(t_1,t_2,\ldots,t_n).\alpha\in S$, but there is no assignments $(c_1/t_1,c_2/t_2,\ldots,c_n/t_n)$ such that $P(c_1,c_2,\ldots,c_n)$ and $\alpha(c_1/t_1,c_2/t_2,\ldots,c_n/t_n)\in S$, then replace S by sets $S\cup$

$\{\alpha(c_1/t_1,c_2/t_2,\ldots,c_n/t_n)\}$ where $(c_1/t_1,c_2/t_2,\ldots,c_n/t_n)$ are all assignments of $(t_1,t_2,\ldots,t_n)$ such that $P(c_1,c_2,\ldots,c_n)$;

Rule 3: if $\alpha(y/b)\in$ S for every constant $b$ such that $R(a,b)$, $\forall R(a,y).\alpha$ is a sub formula of one element of S and $\forall R(a,y).\ \alpha$ does not in S, then replace S by $S'=S\cup\{\forall R(a,y).\alpha\ \}$;

Rule 3′: if $\forall R(a,y).\alpha \in$ S, but there is a constant $b$ such that $R(a,b)$ and $\alpha(y/b)$ is not in S, then replace S by $S'=S\cup\{\alpha(y/b)\}$;

Rule 4: if there is a constant $b$ such that $R(a,b)$ and $\alpha(y/b)\in$ S, $\exists R(a,y).\alpha$ is a sub formula of one element of S but $\exists R(a,y).\alpha$ is not in S, then replace S by $S'=S\cup\{\exists R(a,y).\alpha\}$;

Rule 4′: if $\exists R(a,y).\alpha \in$ S, but there is not a constant $b$ such that $R(a,b)$ and $\alpha(y/b)\in$ S, assume the set of all constant $b$ such that $R(a,b)$ is $\{b_1,b_2,\ldots,b_n\}$, then replace S by $n$ sets $S_i=S\cup\{\alpha(y/b_i)\}(i=1,2,\ldots,n)$;

Rule 5: if $\alpha\in$ S and $\beta\in$ S, $\alpha\wedge\beta$ is an sub formula of one element of S, but $\alpha\wedge\beta$ is not in S, then replace S by $S'=S\cup\{\alpha\wedge\beta\}$;

Rule 5′: if $\alpha\wedge\beta\in$ S but $\alpha$ and $\beta$ are not both in S, then replace S by $S'=S\cup\{\alpha,\beta\}$;

Rule 6:if $\alpha\in$ S or $\beta\in$ S, $\alpha\vee\beta$ is an sub formula of one element of S and $\alpha\vee\beta$ is not in S, then replace S by $S'=S\cup\{\alpha\vee\beta\}$;

Rule 6′: if $\alpha\vee\beta\in$ S but none of the $\alpha$ and $\beta$ is in S, then replace S by $S'=S\cup\{\alpha\}$ and $S''=S\cup\{\beta\}$;

Rule 7: if$\alpha\rightarrow\beta\in$ S and $\alpha\in$ S but $\beta$ in not in S, then replace S by $S'=S\cup\{\beta\}$;

**Definition 3.2** (clash) We say that a primitive formula set S has a clash if $\alpha$ and $\neg\alpha$ are both in S for certain task formula $\alpha$.

To test whether a finite set of primitive task formulas $S_1$ is satisfiable or not, we set $M_1=\{S_1\}$ and apply the transformation rule in the definition 3.3 to M as long as possible then we finally end up with a complete Set $M_r$ i.e. a set to which no rule are applicable. The initial set $S_1$ is satisfiable if and only if there is a set in $M_r$ does not contain a clash (see the follow part of this section for a proof). The test procedure can be defined in pseudo programming language as follows:

**Algorithm 3.1** (satisfiability judgment) The following procedure takes a finite set of primitive formulas as an argument and checks whether it is satisfiable or not.

```
Define   procedure check-satisfiable(S1)
    r:=1;
    M1:={S1}
    While 'a transformation rule is applicable to Mr'
        do
            r:=r+1
            Mr:=apply-a-transformation rule(Mr-1)
        od
    if 'there is an S ∈ Mr that does not contain a clash'
        then return YES;
        else return NO.
```

For example, let {P(room), R(room, mop)} be the domain knowledge. P(room) means that the cleaner is in charge of the room. R(room, mop) means that mop is the necessary tool to clean the room. If the ability specification is

{$\forall P(x).(\forall R(x,y).F(y)\rightarrow C(x))$, F(mop)}, where $\forall P(x).(\forall R(x,y).F(y)\rightarrow C(x))$ means that for every object $x$ that the cleaner is in charge of, if be given all necessity tools the cleaner can accomplish the task of cleaning it. F(mop) means that the keeper can give the cleaner a mop. To judge whether C(room) is accomplishable or not, i.e. whether the cleaner can clean the room or not, we need to judge whether the set $S_1$={$\forall P(x).(\forall R(x,y).F(y)\rightarrow C(x))$, F(mop),$\neg$C(room)} is satisfiable or not. The set $M_i$ generated in the testing process are:

$M_1$={{$\forall P(x).(\forall R(x,y).F(y)\rightarrow C(x))$, F(mop), $\neg$ C(room)}};

$M_2$={{$\forall P(x).(\forall R(x,y).F(y)\rightarrow C(x))$, F(mop), $\neg$C(room),$\forall R(room,y).F(y)\rightarrow C(room)$}}

( by Rule 1 , $\forall P(x).(\forall R(x,y).F(y)\rightarrow C(x)$ )and P(room) );

$M_3$={{$\forall P(x).(\forall R(x,y)F(y)\rightarrow C(x))$, F(mop), $\neg$C(room)}, $\forall R(room,y).F(y)\rightarrow C(room)$, $\forall R$ (room,$y$).F($y$) }

( by Rule 3 , R(room, mop) and F(mop) );

$M_4$={{$\forall P(x).(\forall R(x,y).F(y)\rightarrow C(x))$, F(mop), $\neg$C(room)}, $\forall R(room,y).F(y)\rightarrow C(room)$, $\forall R(room,y).F(y)$, C(room)}}

( by Rule 7, $\forall R(room,y).F(y)\rightarrow C(room)$ and $\forall R(room,y).F(y)$ ).

Then $S_1$ is not satisfiable because there is not a set in $M_4$ that does not contain a clash. So we know that C(room) is accomplishable.

For a primitive task formula $\beta$ the length of $\beta$, denoted by $|\beta|$, is inductively defined as:

1. If $\beta$ is an atomic task formula or $\beta=\neg\alpha$ and $\alpha$ is an atomic task formula, then $|\beta|=1$;
2. if $\beta=\forall P(t_1,t_2,\ldots,t_n).\alpha$ or $\beta=\exists P(t_1,t_2,\ldots,t_n).\alpha$ then $|\beta|=|\alpha|+1$;
3. if $\beta=\forall R(t,y).\alpha$ or $\beta=\exists R(t,y).\alpha$, then $|\beta|=|\alpha|+1$;
4. if $\beta=\alpha\wedge\gamma$ or $\beta=\alpha\vee\gamma$, then $|\beta|=|\alpha|+|\gamma|$;
5. if $\beta=\alpha\rightarrow\gamma$, then $|\beta|=|\alpha|+|\gamma|$.

We call a task formula $\alpha$ that has the form $\alpha=\beta\wedge\gamma$ is a $\wedge$-task. The maximal $\wedge$-expression of a $\wedge$-task is an express $\alpha_1\wedge\alpha_2\ldots\wedge\alpha_n$ such that $\alpha_i$ is no long a $\wedge$-task for every i($1\leq i\leq n$). If $\alpha$ is a $\wedge$-task and its maximal $\wedge$-expression is $\alpha_1\wedge\alpha_2\ldots\wedge\alpha_n$, then the $\wedge$-length of $\alpha$ is $n$. The concept of $\vee$-task and the $\vee$-length of a $\vee$-task, the concept of $\rightarrow$-task and the $\rightarrow$-length of a $\rightarrow$-task all can be defined analogously.

**Proposition 3.1** The algorithm 3.1 can always compute a complete set $M_r$ in finite time and the initial set $S_1$ is not satisfiable if and only all set in $M_r$ contain a clash.

Proof: Because the number of different assignments $(c_1/t_1,c_2/t_2,\ldots,c_n/t_n)$ of $(x_1,x_2,\ldots,x_n)$ such that $P(c_1,c_2,\ldots,c_n)$ for every predict P that occurs in domain knowledge, the number of different sub formulas that has the form $\forall P(t_1,t_2,\ldots,t_n).\alpha$ of elements of $S_1$, the number of different sub formulas that has the form $\exists P(t_1,t_2,\ldots,t_n).\alpha$ of elements of $S_1$, the number of different constant $b$ which satisfies $R(a,b)$ for each pair ($a$,R) of constant and role that occurs in $S_1$, the number of the sub formulas that has the form $\forall R(t,y).\alpha$ of elements of $S_1$, the number the sub formulas that has the form $\exists R(t,y).\alpha$ of elements of $S_1$, the number of the different sub formulas that has the form of $\alpha\wedge\beta$ of elements of $S_1$ and the maximal $\wedge$-length of them, the number of different sub formula that has the form $\alpha\vee\beta$ of elements of $S_1$ and the maximal $\vee$-length of them, the number of the different formula that has the form $\alpha\rightarrow\beta$ of the elements of $S_1$ and the maximal $\rightarrow$-length of them are all finite, then it

can be proved that the rules in definition 3.1 can only be applied for finite times, so the computation process will terminate in finite time.

The second part of the proposition is a consequence of lemma 3.1 and lemma 3.2 bellows, where the notion of contradictory formula set which is syntactic equivalent of not satisfiable formula set is defined by induction on the relation of "descendant". A formula set S occurring in the computation is contradictory with respect to the computation if and only if

1. S does not have a descendant and contains a clash or
2. all descendants of S are contradictory.

**Lemma 3.1** If the initial formula set is contradictory with respect to a given computation then it is not satisfiable.

Proof: The proof is by induction on the definition of contradictory with a case analysis according to the transformation rule applied. Assume $S_1$ is a given set of formulas which is contradictory with respect to a given computation, we will show that it is not satisfiable.

If $S_1$ does not have a descendant, then it must have a clash. Obviously a set of formulas that have a clash is not satisfiable. For the induction step, assume S is satiafiable, we have to show that the descendant (resp. one of the descendant in the case of rule 2′, rule 4′, and rule 6′ ) of S is satisfiable too, this will be contradiction to the induction hypothesis, because all descendants of contradictory set are contradictory.

We shall only demonstrate the case of rule 5. The other cases can be treated similarly. Assume that rule 5 is applied to a set, denoted by $S_{r-1}$, means that there are two formula $\alpha$ and $\beta$ such that $\alpha\in S_{r-1}$ and $\beta\in S_{r-1}$ and the descendant of $S_{r-1}$, denoted by $S_r$, is equal to $S_{r-1} \cup \{\alpha\wedge\beta\}$. If the interpretation I and the assignment $(x_1/c_1,x_2/c_2\ldots,x_n/c_n)$ satisfy $S_{r-1}$, then we have $(\alpha(x_1/c_1,x_2/c_2\ldots,x_n/c_n))^I=1$ and $(\beta(x_1/c_1,x_2/c_2\ldots,x_n/c_n))^I=1$, thus $((\alpha\wedge\beta)(x_1/c_1,x_2/c_2\ldots,x_n/c_n))^I=1$ by the definition of the semantic of closed task formulas, so I and $(x_1/c_1,x_2/c_2\ldots,x_n/c_n)$ satisfy $S_r$ too.

**Lemma 3.2** If the initial formula set is not contradictory with respect to a given computation then it is satisfiable.

Proof: If $S_1$ is not contradictory then there is a primitive formula set $S\supseteq S_1$ in the complete set $M_r$ such that there is no clash in S. Assume the set of all different free variable occur in S is $\{x_1,x_2,\ldots,x_m\}$, we define an interpretation $I=(\Delta^I,\cdot^I)$ as follows:

$\Delta^I$ is the set of all constants. $\cdot^I$ assigns to each constant itself, to each predict P a set $P^I=\{(a_1, a_2, \ldots, a_n)|a_i\in\Delta^I\ (1\le i\le n)$ and $P(a_1, a_2,\ldots, a_n)\}$, to each role R a set $R^I=\{(a,b)|\ a,b\in\Delta^I$ and $R(a,b)\}$. Let $(c_1,c_2,\ldots,c_m)$ be an arbitrary m-tuple of constants, $\cdot^I$ assigns each atomic task $A(a_1,a_2,\ldots,a_j)(j=1,2,3,\ldots)$ an element of $\{0,1\}$ according to following rule:

$$A(a_1,a_2,\ldots,a_j)^I=\begin{cases}0 & \text{if } \neg A(a_1,a_2,\ldots,a_j)\in S(x_1/c_1,x_2/c_2,\ldots,x_m/c_m)\\ 1 & \text{else}\end{cases}$$

We will prove that the interpretation I and the assignment $(c_1/t_1,c_2/t_2,\ldots,c_m/t_m)$ satisfy S, so satisfy $S_1$ also. We use induction on the length of $\alpha$.

If $|\alpha|=1$, then $\alpha$ is an atomic task or $\alpha=\neg\beta$ and $\beta$ is an atomic task. Then $(\alpha(x_1/c_1,x_2/c_2\ldots,x_m/c_m))^I=1$ by the definition of I.

Assume $|\alpha|=k$ ($k>1$), we prove that $(\alpha(x_1/c_1,x_2/c_2\ldots,x_m/c_m))^I=1$. In the case of $\alpha=\forall P(t_1,t_2,\ldots,t_n).\beta$. $M_r$ is complete then we have $\beta(t_1/a_1,t_2/a_2,\ldots,t_n/a_n)\in S$ for every assignment $(t_1/a_1,t_2/a_2,\ldots,t_n/a_n)$ such that $P(a_1,a_2,\ldots,a_n)$, so $(\beta(t_1/a_1,t_2/a_2,\ldots,t_n/a_n)(x_1/c_1,x_2/c_2\ldots,x_m/c_m))^I=1$ by the induction hypothesis, so we have $(\alpha(x_1/c_1,x_2/c_2\ldots,x_m/c_m))^I=1$. The other cases can be proved analogously.

## 4 Logic DTL

The logic *DTL* (Description Logic of Tasks) that we are going to define in this section is intended to axiomatize the set of accomplishable task formulas. It will be mentioned that the concept of quasiaction and quasireaction of a task formula is same as those in [2].

**Definition 4.1** (*DTL*). The axioms of *DTL* are all the primitive formulas that are accomplishable.

The rules of inference are

A-rule:

$\frac{\pi}{\alpha}$, where $\pi$ is an elementary quasiaction for $\alpha$.

R-rule：

$\frac{\overline{\alpha},\pi_1,\pi_2,\cdots,\pi_e}{\alpha}$, where $e\geq 1$ and $\pi_1,\pi_2,\cdots,\pi_e$ are all quasireaction for $\alpha$, $\overline{\alpha}$ is the primitivization of $\alpha$.

**Theorem 4.1**(soundness) Let $\alpha$ be a task formula, if DTL $\vdash\alpha$ then $\alpha$ is accomplishable.

**Theorem 4.2** (completeness) Let $\alpha$ be a task formula, if $\alpha$ is accomplishable, then DTL $\vdash\alpha$.

The only different between the logic DTL and the logic L proposed in [2] is the different of their axioms. Axioms of DTL are primitive task formulas that are accomplishable while the axioms of L are formulas provable in classical first order logic. The rules of inference in them are same. Keep in mind that the concept of strategy, quasiaction and quasireaction used in this paper are same to those in [2]. So, the proof of the soundness, completeness of DTL can be carried out analogically as the proof of the corresponding properties of the logic L.

**Theorem 4.3** (decidability) DTL is decidable.

Proof: Here is an informal description of a decision procedure for DTL $\vdash\alpha$, together with a proof, by induction on the additive complexity of$\alpha$, that the procedure takes a finite time. Given a formula $\alpha$

(a) If $\alpha$ is primitive, then we can judge whether it is an axiom, i.e. whether it is accomplishable using algorithm 3.1 in finite time by Proposition 3.1.

(b) If $\alpha$ is not primitive, then the only way it can be proved in DTL is if either one of the elementary quasiactions for it is provable, or all of the elementary quasireactions for it, together with its primitivization, are provable in DTL. Whether the primitivization is provable can be checked in a finite time. Also, as we noted, the

number all the elementary quasiactions and quasireactions for $\alpha$ is finite. So, check each of them for provability in DTL. If it turns out that either one of the elementary quasiactions, or all of the elementary quasireactions together with the primitivization of $\alpha$ are provable in DTL, then output "yes", otherwise output "no". The additive complexities of those elementary quasiactions and quasireactions are lower than the additive complexity of $\alpha$ and, by the induction hypothesis, their provability in DTL can be checked in a finite time. So that this step, too, can be completed in a finite time.

## 5 Conclusion

The description logic of tasks enable tasks description and have more reasoning power, it can be used to structure the cooperation plan system for multi agent system. For example, for behavior modeling of large-scale battlefield simulation, we use it to describe the ability knowledge of military entities and the relations among them, then we can use the reasoning power of it for cooperation actions plan and verification.

## References

1. V.Dignum, J.J. Meyer, F. Dignum, H. Weigand. Formal Specification of Interaction in Agent Societies. In: M. Hinchey, J. Rash, W. Truszkowski, C. Rouff, D. Gordon-Spears (Eds.): Formal Approaches to Agent-Based Systems (FAABS), Lecture Notes in Artificial Intelligence, Springer-Verlag, Volume 2699/2003.
2. Giorgi Japaridze, The logic of tasks, Annals of Pure and Applied Logic 117 (2002).
3. Peep Küngas Analysing AI Planning Problems in Linear Logic – A Partial Deduction Approach Lecture Notes in Computer Science Volume 3171/2004.
4. Peep Küngas, Mihhail Matskin Linear Logic, Partial Deduction and Cooperative Problem Solving Lecture Notes in Computer Science Volume 2990/2004.
5. G.Japaridze, Introduction to computability logic. Annals of Pure and Applied Logic, vol. 123 (2003).
6. A. Blass, A game semantics for linear logic, Ann. Pure Appl. Logic 56 /1992.
7. F. Baader etc. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge, Cambridge University Press 2002.
8. F. Baader and U. Sattler. Tableau Algorithms for Description Logics ， Lecture Notes In Computer Science， Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods ,2000.
9. F. Baader and P. Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. DFKI Research Report RR-91-10, Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, 1991.
10. S. Russel, P. Norwig, Artificial Intelligence: A Modern Approach, Prentice-Hall, Englewood Cli9s, NJ,1995.

# A Logic Programming Formalization for Circumscription

Masahiko Arai

Dept. of Information and Communication Technology, Tokai University
317 Nishino, Numazu, Shizuoka, 410-0395, Japan
arai@wing.ncc.u-tokai.ac.jp

**Abstract.** This paper proposes a practical way for circumscription. The meanings of the practical way are 1) a goal-oriented prover is given for solving circumscription problems. A feature of the prover is that the priority and the variable predicates are suggested so as to prove the intended results whereas usual methods prove queries by giving the priority and the variable predicates in advance. This prover is an application of the one for DNF formulae in the object system T and is the set of meta-rules of a meta-predicate which represents a clause. Circumscription formulae are represented as the meta-rules. The SLD-resolution procedures for the meta-system are given and a circumscription problem is solved by showing an empty node in the SLD-tree if semi-decidable. 2) Based on the prover, formulae satisfying circumscription formulae are given for predicates with functions. Practically circumscription is applied to prove the negation of a predicate p. For the formulae given above to be false, the condition is needed that T made p false is consistent with T. 3) Variable predicates are generalized so as to prove T made p false. By the generalization it is shown that circumscription problems are practically solved by giving a model to show the consistency for the undecidable cases. By encapsulating the consistency check as an oracle a practical logic programming for circumscription is given for queries without universal quantifiers. The oracle for queries with universal quantifiers is also given.

## 1 Introduction

Nonmonotonic logics are important for inferences based on defeasible assumptions in AI [1]. Major nonmonotonic logics are circumscription [7], default logic, and autoepistemic logic. These logics deal with the concepts of plausibility and normality and the proof systems are complicated. As seen in the sequent calculi for nonmonotonic logics these proof systems include both the proof and disproof procedures [3, 5]. Default and autoepistemic logics need the disproof-procedures to generate the extensions and the expansions, respectively. Regarding the provers circumscription is attractive from the following two reasons. One is that in propositional case the complexity of circumscription is $\Pi^p_2$ of polynomial hierarchy whereas that of autoepistemic logic is $\Pi^p_3$-complete for skeptical reasoning [4]. The other is that circumscription is formalized in classical logic by a formula in second-order logic, and resolution procedures can be applied to the provers. The proof-

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 23-32*

systems of circumscription are usually based on minimal modes, and disproof-procedures are needed to generate minimal models. In the tableau calculus it is needed to show that the branches which are not closed are not minimal models by showing that there are predicates not proved in the branches [8]. For a given theory T and a query the MILO-resolution needs to generate a tree which shows that T and the query are consistent, and to prove all the leaves in T [10].

There are several restrictions in these provers, 1) The selection priority of the predicates circumscribed and the variable predicates must be given a priori [6]. 2) There are no procedures for the cases where the predicates include functions, i.e., the ones undecidable. Originally nonmonotonic systems are required to realize the concepts of plausibility and normality needed. The priority and the variable predicates must be chosen such that the requirements are satisfied. Therefore these vary accordingly to the queries. In this meaning a goal-oriented prover is practically needed such that both the circumscribed (the priority) and the variable predicates can be suggested in the processes of proofs. Regarding 2) the circumscription formula includes at least $\Pi^0_1$ formulas of arithmetical hierarchy, and the system is undecidable. Therefore semi-decidable provers are impossible. However if the undecidability is confined it is possible to make provers semi-decidable except the confined undecidable procedures. This method gives a practical way to solve problems for circumscription. For example if the undecidability is due to the proof of the consistency then it can be resolved practically by giving a model though not formally.

This paper proposes a goal-oriented prover which solve practically the above two problems. The proof-system is an application of that for the set of formulae in the disjunctive normal forms (DNF) [2]. The system is the set of Horn clauses called *meta-rules* of a second-order predicate *Prov* called a *meta-predicate* which means a clause. The system is called a *meta-system*, and is generated from the DNF formulae in the object system T. SLD-resolution procedures are given for the meta-system, and the search space is an SLD-tree. The meta-system is sound and complete in the meaning that if Prov with the empty value is proved in the meta-system then the object system is inconsistent, and vice versa. Let T be the set of clauses. For a given predicate p the circumscription formula of p is that $\alpha \leftarrow p$ is proved from $p \leftarrow \alpha$ and $T(p/\alpha)$, where $T(p/\alpha)$ is given by substituting $\alpha$ for p in T. $\alpha$ is called a *circumscribing formula* of p. The circumscription formula is a DNF formula, since the body consists of $p \leftarrow \alpha$ and $T(p/\alpha)$, where $T(p/\alpha)$ is a conjunction of clauses due to the fact that T is the set of clauses. Therefore the circumscription formula is a meta-rule in the meta-system and is called *the circumscription meta-rule*. Generally (T+circumscription) is undecidable and so is the meta-system.

In Sec.2, the proof system is formulated. In Sec.3, based on the prover, circumscribing formulae are given without variable predicates for the set of typical clauses of predicates with functions. Practically circumscription is applied to prove the negation of p. For the formulae given above to be false, the condition must be satisfied that T made p false is consistent with T. When the condition is satisfied, variable predicates are generalized so as to prove T made p false and are called *generalized variable-predicates*. Generally the consistency check is undecidable. By encapsulating the consistency check as an oracle, a practical logic programming for circumscription is given for queries without universal quantifiers, i.e., circumscription

problems are practically solved by giving a model to show the consistency. The oracle for queries with universal quantifiers is also given.

## 2 Formalization

Let (Γ) be a set of formulas of first-order logic. Without loss of generality it is assumed that predicates are one variable except the special predicate =, the equality. Assuming Skolem's functions and the Henkin theory of (Γ), let Γ be the matrix of (Γ) represented as the set of DNF formulae. In the following each upper case letter, U, W, X, Y or Z, takes the values of disjunctions, conjunctions of positive predicates, or the empty symbol □, and each lower case letter except x and y represents a predicate in the formula the upper case letter denotes. x or y is for the variable of (Γ). X and Y are the variables of the meta-system called M whose values are the conjunctions and the disjunctions of the Herbrand base of Γ, respectively. Now introduce a meta-predicate Prov(X;Y) which means Y←X, where X and Y are a conjunction and a disjunction of positive predicates, respectively. When Y={Y1,Y2} and X=X1&X2, Prov(X;Y) is also represented as Prov(X1,X2;Y1,Y2), where & is the conjunction sign and {} means the disjunction of the elements. The meta-system for to Γ is defined as follows.

**Definition 1** Let Γ be the set of the formulas, (Z←W)←(Z1←W1)&…&(Zm←Wm) and (Z'←W')←. The meta-system M is the set of meta-rules as follows.

M0 Prov(X,X1;Y1,Y) ← Prov(X,X1,X1;Y1,Y1,Y),
M1 Prov(X,u;u,Y) ←,
M2 Prov(X;Y) ← Prov(X;u1,Y)&…&Prov(X;uj,Y)&Prov(X,U;Y), U=u1&…&uj,
M3 Prov(X,W;Z,Y) ← Prov(X,W,W1;Z1, Z,Y)&…&Prov(X,W,Wm;Zm, Z,Y),
M4 Prov(X,W';Z',Y) ←,

where u, u1,…, and uj are predicates (x is not shown explicitly). The left and the right formulae of a meta-rule are called the head and the body, respectively. When the negation of a query is added to Γ, the query is represented as a conjunction in the normal form and the corresponding meta-rule is M3 with W and Z empty. The body of a meta-rule without the head is called a *goal clause* for simplicity. It is noted that *M is a proving system without the negation symbol ~.*

**Definition 2 (The SLD-tree)** The root node is Prov(□; □). Each node is labeled a conjunction of Prov. Let Prov(X1;Y1) be the left most of the conjunction labeled to a node. Then the conjunctions below the node are generated as follows. In the following 'X1 (Y1) includes W (Z)' means that every predicate in W (Z) matches with a predicate in X1 (Y1) with the most general unifier. X-X1 means the formula removed atoms in X1 from X.

P1: If Prov(X1;Y1) matches with the head of M1or M4 then remove it.
P2: If X1 and Y1 include W and Z in M3, respectively, then remove it and add Prov(X1,Wi;Zi,Y1), (1≤i≤m).
P3: If Y1 includes Z in M3 and does not include any ui, (1≤i≤j), where u1&…&uj=

W-X1, then remove Prov(X1;Y1) and add Prov(X1;ui,Y1), (1≤i≤j), and Prov(X1,W,Wi;Zi,Y1), (1≤i≤m).

P4: If Y1 includes Z' in M4 and does not include any ui, (1≤i≤j), where u1&…&uj= W'-X1, then remove Prov(X1;Y1) and add Prov(X1;ui,Y1), (1≤i≤j).

The generated meta-predicates obtained by applying the above procedures repeatedly are called *the descendants* of Prov(X1;Y1). It is noted that for P2, P3, or P4 to be applied to Prov(X1;Y1), Y1 must include Z or Z'. The branch whose leaf is the empty node is called a *success-branch*. Theorems 3 and 4 have been proved in [2].

**Theorem 3** *If there is the empty node in the SLD-tree for M then Γ is inconsistent and vice versa.*

**Theorem 4** *If the SLD-tree includes the empty node then there is a success-branch such that any meta-predicate in a node doesn't appear in the descendants, i.e., there are no loops such that a meta-predicate is expanded repeatedly.*

**Example 5** To show the existence of x for p(x) from {p(a),p(b)}, where a and b are constant. Γ is {p(a),p(b)}← and the negation of the query is ←p(x). Corresponding meta-rules and the goal clause for Prov(□; □) are, respectively,

Prov(X; p(a),p(b),Y) ←, (1)

Prov(X;Y) ← Prov(X;p(x),Y), (2)

← Prov(□; □), (3)

where the other meta-rules, M0, M1, and M2 are omitted (so are hereafter). (2) is the meta-rule corresponding to the negation of the query in Γ. The SLD-tree is

Prov(□;□) - Prov(□;p(x)) - Prov(□;p(x),p(x')) - □.

The second is obtained by matching Prov(□;□) with the head of (2) and P2. The third is given by matching Prov(□;p(x)) with the head of (2) and P2. By Theorem 4, x and x' are different. The last is shown by (1) and P1 with x=a and x'=b.

Let T be the set of clauses. The formulation of circumscription is given in second-order logic with the universal quantifier. The universal quantifiers in first and second-order logics satisfy the same inference rules [9] and a circumscription formula is a DNF formula in T. Therefore the meta-rule for circumscription is in the form of M3 and is given as follows.

C1 Prov(X,p(x);α(x),Y) ← Prov(X,p(x),α(c);p(c),α(x),Y)&T(p/α,x/c':X,p(x);α(x),Y),

where c and c' are constant symbols not appearing in T and T(p/α,x/c':X,p(x); α(x),Y) is the conjunction of meta-predicates for clauses in T replaced p and x with α and c', respectively. For example when T is the set of q(x)←p(x) and r(x)←s(a),

T(p/α,x/c':X,p(x);α(x),Y)≡Prov(X,p(x),α(c');q(c'),α(x),Y)

& Prov(X,p(x),s(a);r(c'),α(x),Y).

However the second meta-predicate is removed since it is true in M.

The other meta-rules needed for (T+circumscription) relate to the equality (=).

C2 Prov(X,X1(x);Y1(x),Y) ← Prov(X,X1(x),X1(y),x=y;Y1(y),Y1(x),Y)
&Prov(X,X1(x);x=y,Y1(x),Y),

C3 Prov(X,x=y,x=y';Y) ← Prov(X,y=y';Y),

C4 Prov(X;x=x,Y) ←,

where the variables x and y in X1 and Y1 are explicitly shown. C2 is the meta-rule to show that if a meta-predicate is proved at x=y and for any values not equal to y then the meta-predicate is proved for any values of x.

It is noted that α in C1 is a variable in the meta-system. By definition α(x) in the head of C1 is a disjunction of positive predicates and α(c) in the first meta-predicate in the body is a conjunction of positive predicates. Moreover α is unified with any formula. This is understood by regarding α as a positive predicate with the following auxiliary meta-rules, respectively, corresponding to {α1(x),α2(x)}, α1(x)&α2(x), {~α1(x),α2(x)}, and ~α1(x)&α2(x) for α(x).

A1 Prov(X, α(x);Y) ← Prov(X,α1(x);Y)&Prov(X, α2(x);Y),

A2 Prov(X;α(x),Y) ← Prov(X;α1(x),Y)&Prov(X;α2(x),Y),

A3 Prov(X;α(x),Y) ← Prov(X,α1(x);α2(x),Y),

A4 Prov(X,α(x);Y) ← Prov(X,α2(x);α1(x),Y).

**Definition 6** *The meta-system MC* for (T+circumscription) is the set of M0, M1, M2, C1, C2, C3, C4 and the meta-rules of M4 for the clauses in T, and the auxiliary meta-rules A1, A2, A3, and A4.

## 3 Practical Logic Programming for Circumscription

**Example 7** T is {p(a),p(b)}←. Then p(x)={p(a)&x=a, p(b)&x=b} is proved in MC. p(x)←{p(a)&x=a, p(b)&x=b} is obvious. {p(a)&x=a, p(b)&x=b}←p(x) is shown as follows. Meta-rules are with α(x)≡{p(a)&x=a,p(b)&x=b} and a constant d not in MC,

Prov(X;p(a),p(b),Y) ← ,

Prov(X;Y) ← Prov(X,p(d);α(d),Y).

A success-branch is given as follows, with X1≡p(d)&α(c), and Y1≡{p(c),α(d)}. The root, Prov(□;□), is omitted (so is hereafter).

Prov(p(d);α(d)) - Prov(X1;Y1) - Prov(X1,α(a),c=a;p(a),Y1)&Prov(X1;c=a,Y1) -
- Prov(X1;c=a,Y1) - Prov(p(d),α(c);c=a,c=b,p(c),α(d)) - □.

The second node is given by matching with C1 and is the first meta-predicate in the body of C1 since T(p/α,x/c':X,p(x); α(x),Y) is true. The third is given by applying C2 with y=a. By applying A1 to the first meta-predicate Prov(X1,p(a),c=a;p(a),Y1) and Prov(X1,p(b),c=b,c=a;p(a),Y1) are obtained. Both are true by P1 and by C3 with a=b false, respectively, and the fourth node is given. Similarly the fifth node is obtained by applying C2 with y=b. The last is given by A1 and P1.

It is noted that {x=a,x=b}←p(x) is also proved in Example 7. Then the following two circumscription meta-rules corresponding to α(x)≡x=a and α(x)≡x=b are used.

Prov(X,p(x);x=a,Y) ← Prov(X,p(x),c=a;p(c),x=a,Y),

Prov(X,p(x);x=b,Y) ← Prov(X,p(x),c'=b;p(c'),x=b,Y).

Similarly the well-known solution, i.e., for all x and all y {(x=a←p(x)), (y=b←p(y))}, is also proved in MC by using the above two circumscription meta-rules with the following meta-rule for the query and constants, d and d', not in MC.

Prov(X;Y) ← Prov(X,p(d),p(d');d=a,d'=b,Y).

**Example 8 (The priority and variable predicates)** With A, S, E, p1, and p2 for adult, student, employed, abnormal1, and abnormal2, respectively, Let T be p1(x)←S(x)&E(x), {E(x),p2(x)}←A(x), A(x)←S(x), S(m)← and let the query be ~E(m), where m (Mary) is a constant. The corresponding meta-rules are, respectively,

Prov(X,S(x),E(x);p1(x),Y) ←, (4)

Prov(X,A(x);E(x),p2(x),Y) ←, (5)

Prov(X,S(x);A(x),Y) ←, (6)

Prov(X;S(m),Y) ←, (7)

Prov(X;Y) ← Prov(X,E(m);Y). (8)

The circumscription meta-rules for p1 and p2 are, respectively,

Prov(X,p1(x);α1(x),Y)
←Prov(X,p1(x),α1(c);p1(c),α1(x),Y)&Prov(X,p1(x),S(c'),E(c');α1(c'),α1(x),Y), (9)

Prov(X,p2(x);α2(x),Y)
←Prov(X,p2(x),α2(d);p2(d),α2(x),Y)&Prov(X,p2(x),A(d');E(d'),α2(d'),α2(x),Y).(10)

It is noted that the second meta-predicate in the body of (9) is T(p1/α1,x/c':X,p1(x); α1(x),Y) since from (5) to (8) the meta-predicates are not changed and are true. Therefore these meta-predicates in T(p/α,x/c':X,p(x); α(x),Y) are dropped. Similarly (10) is obtained. The first and the second nodes of a success-branch are given by making α1(x) empty for a value x' in (9),

Prov(E(m); □) -
- Prov(E(m);p1(x'))&Prov(p1(x'),α1(c);p1(c))&Prov(p1(x'),S(c'),E(c');α1(c')) -.

The second node is given by P3. The first meta-predicate of the second node is expanded into Prov(E(m);S(x'),p1(x'))&Prov(E(m);E(x'),p1(x')) by matching with (4) and by applying P4 . From (7) the first meta-predicate is removed with x'=m by P1, and the second meta-predicate is also removed by P1. Therefore the first meta-predicate in the second node is removed. The third and the last nodes are

- Prov(p1(m),α1(c);p1(c))&Prov(p1(m),S(c'),E(c');α1(c')) - □.

The first meta-predicate in the third node is matched with (4) by unifying x and α1(c) with c and S(c)&E(c), respectively, and is removed. By using the auxiliary meta-rule A2, it is shown that the second meta-predicate is also removed by P1 and the empty node is obtained. The condition that α1(m) is false is satisfied by requiring that S(x)&E(x) is false, i.e., there are no students employed. It is easily shown that the empty node is not obtained by making α2(m) empty. Therefore the priority of p1 is higher than that of p2 and it is required that S or E is the variable predicate.

Suppose that for another student k (Ken), E(k) is another plausible query. Adding

Prov(X;S(k),Y) ←, (7')

require that Prov(□;E(k)) is proved. A success-branch is by making α2(x) in (10) empty at x=x',

Prov(□;E(k)) -
- Prov(□;E(k),p2(x'))&Prov(p2(x'),α2(d);p2(d))&Prov(p2(x'),A(d');E(d'),α2(d')) -.

The first meta-predicate of the second node is replaced by Prov(□;A(k),E(k),p2(k)) by matching with (5) for x'=k and from P4. From (6) with P4 and (7') with P1 the first meta-predicate is removed. The third and the last nodes are

- Prov(p2(k),α2(d);p2(d))&Prov(p2(k),A(d');E(d'),α2(d')) - □.

The first meta-predicate of the third node is removed by applying A4 with α2(d)=~α21(d)&α22(d) and by matching with (5) by unifying α21(d) and α22(d) with E(d) and A(d), respectively. The second meta-predicate is also removed by applying A2 and A3. The priority of p2 is higher than that of p1 and the variable predicate is A or E. The condition that α2(k) is false is satisfied by requiring that ~E(k)&A(k) is false. It is easily shown that p1(k) and p2(m) are true, i.e., Ken is abnormal as a student and so is Mary as an adult. In this case the condition that S(x)&E(x) is false is not correct since α1(k) is true. An alternative is that α1(x)≡S(x)&E(x)≡x=k, i.e., only

Ken is abnormal as a student. Similarly $\alpha 2(x) \equiv \sim E(x) \& A(x) \equiv x=m$, i.e., only Mary is abnormal as an adult.

Similarly the following Example 9 is proved in MC. Let Q(x) be a conjunction of positive or negative predicates except p and if Q(x) includes p then p is positive with functions.

**Example 9** Let T be $p(x) \leftarrow Q(x)$, or $\{p(x),p(a)\} \leftarrow Q(x)$, or $\{p(x),p(f(x))\} \leftarrow$. Then $p(x)=Q(x)$, or $p(x)=\{Q(x)\&\sim p(a),\ p(a)\&x=a\}$, or $p(x)=p(x)\&\{p(f(-2,x)),p(f(2,x))\}$, respectively, is proved in MC.

As seen in Examples 8 and 9, to prove $\sim p(t)$ for a given term t, a sufficient condition is that T made p(t) false is required. Generally this form of circumscription is obtained by using generalized variable-predicates defined below.

**Definition 10** Let S be a subset of the *Herbrand universe* of T. Then *T([p/□,S])* is defined by the set of clauses of the form $Z \leftarrow W$ in T for which W doesn't include p(t) and Z includes p(t) and from which p(t) is removed for t in S. *Generalized variable-predicates* are defined such that T([p/□,S]) is satisfied when T([p/□,S]) is consistent with T.

**Theorem 11** *If (T+T([p/□,S])) is consistent then ~p(t) for t in S is proved with the generalized variable-predicates.*

**Proof:** Let $\alpha(x)$ be false for x in S, and be p(x) for x not in S. Then $p(x) \leftarrow \alpha(x)$ is proved and $T(p/\alpha)$ is proved assuming T([p/□,S]).

It is noted that for c not in T, $\sim p(c)$ is not proved since $\alpha$ is a model of p, t is in the Herbrand universe of T. It is also noted that ~p is not proved from T([p/□,S]), but is proved by circumscription with generalized variable-predicates. Example 7 is also shown by Theorem 11. Because let S be the set of x such that $x \neq a,b$. Since T([p/□,S]) is empty and consistent with T, without generalized variable-predicates, $\alpha(x)$ is given by the one which is false for $x \neq a,b$ and is p(a) for x=a and p(b) for x=b.
Usually the consistency check is undecidable. By encapsulating the consistency check as an oracle the logic programming with oracles is given in the following.

**Definition 12** For a term t, *O(p(t):S)* is the oracle answering true if T([p/□,S']) is consistent with T and false otherwise, where S' is the sum of S and t. The extended predicate $Prov(X;Y:\pi,\Sigma)$ is defined from Prov(X;Y) by adding two variables $\pi$ and $\Sigma$ for predicates and subsets of the region of x, respectively. The oracle and *Prov(X;Y:π,Σ)* satisfy the following meta-rules O1, O2, and G for the initial goal clause.

O1 $Prov(X,r(x);Y:r(x),\Sigma) \leftarrow Prov(X,r(x);O(r(x):\Sigma),Y:r(x),\Sigma)$,

O2 $Prov(X;Y:r(x),\Sigma) \leftarrow Prov(X;Y:\pi,\Sigma')$,

G $\leftarrow Prov(□;□:\pi,\Sigma)$,

where r is the variable for predicates and Σ' is defined by adding x to Σ.

O2 is for preserving the information regarding Σ obtained in the proof. The following Theorem 13 is immediately obtained.

**Theorem 13** *Let Mo be the meta-system M for T with O1 and O2. If ~p(t) is proved by using C1 once then ~p(t) is proved in Mo, and Mo is decidable regarding circumscription. The converse is obtained for MC with generalized variable-predicates.*

**Example 14** Let T be p(a)← and p(x)←p(f(x)). Suppose that there is not an n such that a=f(n,b). Let S be the set of (b, f(b), ..., f(n,b), ...). Then T([p/□,S]) is empty. Therefore ~p(b) is proved without generalized variable-predicates. In Mo there is the success-branch due to the oracle.

Prov(p(b);□) - Prov(p(b);O(p(b):S)) - □.

It is noted that the oracle in Definition 12 is for queries with the existential quantifier. For the query, α(x)≡f(n(x),x)=a, where n(x) is the Skolem's function for n, the query includes the universal quantifier regarding x. In this case the oracle requires the information about the region in which α is false and is more complicated than the one give above.

**Definition 15** Let *O(X1,p;Y1:Sp)* is the oracle answering true if T([p/□,Sp]) is consistent with T and false otherwise, where Sp is the complement of the region in which Y1(x)←p(x)&X1(x) is proved. The oracle satisfies

Ou Prov(X,X1(x),r(x);Y1(x),Y:π,Σ) ← Prov(X;O(X1,r;Y1:Sr),Y:π,Σ),

which is proved by Theorem 11. By using Ou, with α(x)≡f(n(x),x)=a, α(x)←p(x) is proved since there is a success-branch such that

Prov(p(c);α(c):π,Σ) - Prov(□;O(p;α:Sp):π,Σ) - □,

where Sp is x such that f(n,x)≠a for all n. The empty node is given by the oracle since T([p/□,Sp]) is consistent with T.

As is easily seen when T is {p(a),p(b)}←, ~p(a) or ~p(b) can be inferred with generalized variable-predicates. Therefore there is a problem in the definition of generalized variable-predicates. To remove the problem one way is to require that T([p/□,S]) is empty. However it is well known that meaningful results aren't given under the condition. Another way is to restrict the inference regarding ~p(a) or ~p(b) as follows.

**Definition 16** For a generalized variable-predicate p(t) let φ be a disjunction of positive or negative predicates. Consider the restriction that ~p(t) can't be inferred if there is a clause φ such that {p(t),φ} is proved but φ is not proved in T. *'Semi-general'* and *'restricted'* variable-predicates are the cases where φ is the disjunction of positive p and φ is any disjunction not including ~p(t), respectively.

It is easily shown that for the semi-general case the proof of ~p doesn't depend on S but depends on another circumscribed predicate q is used in the proof. For the restricted case it is also shown that the necessary and sufficient condition to prove ~p(t) is that there are no clauses in T including p(t). An easy way to implement the above restrictions is the use of oracles answering under the restrictions. Then the oracle for the semi-general case is the most undecidable among the three.

## 4 Conclusion

A goal-oriented prover for solving circumscription problems was presented. A feature of the prover is that the priority and the variable predicates are suggested so as to prove the intended results. Based on the prover, formulae were given which satisfy circumscription formulae without variable predicates for the set of typical clauses of predicates with functions. Practically circumscription is applied to prove the negation of a predicate. For the formulae given above to be false, the condition must be satisfied that T made p false is consistent with T. Variable predicates are generalized so as to prove T made p false. By the generalization it was shown that circumscription problems are practically solved by giving a model to show the consistency for the cases where predicates include functions for which the problems become undecidable. Generally consistency problems are undecidable. By encapsulating the consistency check as an oracle a decidable prover was presented for queries without universal quantifiers. The oracle with universal quantifiers was also given. This prover is practical in the meaning that the consistency is proved by giving a model of T. The restrictions, semi-general and restricted, were considered for generalized variable-predicates not to infer undesirable predicates. These restrictions are additive and more unified formulations are desired.

## References

1. G. Antoniou. Nonmonotonic Reasoning. The MIT Press, 1997.
2. M. Arai. A Parallelism-Oriented Prover with a Meta-Predicate. In *Proceedings of Second IEEE International Conference on Intelligent Systems,* pages 138-143. IEEE, 2004.
3. P. A. Bonatti and N. Olivetti. Sequent Calculi for Propositional Nonmonotonic Logics. *Transactions on Computational Logic,* 3(2):226-278. ACM, 2002.
4. M. Cadoli and M. Schaerf. A Survey of Complexity Results for Non-Monotonic Logics. *The Journal of Logic Programming,* 17:127-160. Elsevier Science Publishing, 1993.
5. U. Egly and H. Tompits. Proof-Complexity Results for Nonmonotonic Reasoning. *Transactions on Computational Logic,* 2(3):340-387. ACM, 2001.
6. V. Lifschitz. Computing Circumscription. In *Proceedings of IJCAI-85,*121-127, 1985.
7. J. McCarthy. Applications of Circumscription to Formalize Commonsense Knowledge. *Artificial Intelligence,* 28:89-116, 1986.
8. I. Niemela. Implementing Circumscription Using a Tableau Method. In *Proceedings of ECAI 96.* pages 80-84, John Wiley & Sons,1996.
9. W. Pohlers. Subsystems of Set Theory and Second Order Number Theory. *Handbook of Proof Theory (S. R. Buss, Editor)*, pages 209-336, Elsevier Science B. V., 1998.
10. T. C. Przymusinski. An Algorithm to Compute Circumscription. *Artificial Intelligence,* 38:49-73, 1989.

# Constraint Satisfaction

# Genetic Algorithms for Dynamic Variable Ordering in Constraint Satisfaction Problems

H. Terashima-Marín, R. de la Calleja-Manzanedo, and M. Valenzuela-Rendón

Center for Intelligent Systems, Tecnológico de Monterrey
Ave. Eugenio Garza Sada 2501 Sur, Monterrey, Nuevo León 64849 Mexico
{terashima@itesm.mx,rlight_renecm@hotmail.com,valenzuela@itesm.mx}

**Abstract** A Constraint Satisfaction Problem (CSP) can be stated as follows: we are given a set of variables, a finite and discrete domain for each variable, and a set of constraints defined over the values that each variable can simultaneously take. The objective is to find a consistent assignment of values to variables in such a way that all constraints are satisfied. To do this, a deterministic algorithm can be used. However, the order in which the variables are considered in the search process has a direct impact in the efficiency of the algorithm. Various heuristics have been proposed to determine a convenient order, which are usually divided in two types: static and dynamic. This investigation in particular uses Genetic Algorithms as a heuristic to determine the dynamic variable ordering during the search. The GA is coupled with a conventional CSP solving method. Results show that the approach is efficient when tested with a wide range of randomly generated problems.

## 1 Introduction

A Constraint Satisfaction Problem [1] (CSP) is composed of a finite set of variables, a discrete and finite domain of values for each variable, and a set of constraints specifying the combinations of values that are acceptable. The aim is to find a consistent assignment of values to variables in such a way that all constraints are satisfied, or to show that a consistent assignment does not exist. Several deterministic methods exist in the literature to carry out this process [2,1], and solutions are found by searching systematically through the possible assignments to variables, usually guided by heuristics. Many investigations have shown that the order in which the variables are considered for instantiation in the search has a direct impact in its efficiency [3]. There is a wide range of practical problems that can be modeled as CSPs. Applications of the standard form of the problem have included theorem proving, graph coloring and timetabling, machine vision, and job-shop scheduling [1]. Various heuristics have been proposed in the literature to determine an appropriate variable ordering, which can be classified in two types: static and dynamic. The heuristics of Static Variable Ordering (SVO) generate an order before the search begins, and it is not changed thereafter. In the heuristics of Dynamic Variable Ordering (DVO), the order in which the next variable to be considered at any point depends on the current


*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 35-44*

state of the search. It has been observed that heuristics for DVO outperform those heuristics for SVO [4,3]. This article presents an investigation which uses a Genetic Algorithm (GA) [5] as a dynamic heuristic to determine the appropriate variable ordering during the search. The GA is used with a Forward Checking algorithm (FC) and in this scheme the FC algorithm calls the GA which decides the next variable (one or more) to be instantiated. Results of this approach are compared against three other heuristics that have been widely used in similar studies and have provided reasonable performance for a variety of problems.

The reminder of this article is organized as follows. The next section describes the proposed solution model. Section 3 presents the results obtained and their discussion when the model is tested over different instances of CSPs. Finally, in Section 4 the conclusions are included.

## 2 Methodology

This report presents a combination of aspects of Constraint Satisfaction and Evolutionary Computation. This association has been used before. For instance, recent work by Craenen et al. [6] presents a comparative study on the performance of different evolutionary algorithms for solving CSPs. Research by Eiben [7] also discusses a methodology and directions for developing hybrid approaches with both techniques. The work presented in this paper, however, establishes the connection in a different way by concentrating on the problem of dynamic variable ordering when solving constraint satisfaction problems.

We herein describe a model to define the instances of CSP problems used in this work; they are binary CSPs (problems in which the constraints involve only two variables) defined by a four-tuple $\langle n, d, p_1, p_2 \rangle$, where $n$ is the number of variables, $d$ is the domain associated with each variable (for this investigation it is the same for all variables), $p_1$ is the probability that there is a constraint between a pair of variables, and $p_2$ the probability that, given that there is a constraint between two variables, the pair of values is inconsistent. This means that $p_1$ and $p_2$ represent an approximation of constraints in the problem (*constraint density*) and a number of inconsistent pairs of values (*constraint tightness*), respectively. A problem of this kind will have $p_1 \frac{n(n-1)}{2}$ constraints, and $p_2 d^2$ over each constraint. The same model has been used in other similar studies [8,9,10].

As a basis for comparison, this work uses several variable ordering heuristics that have been previously studied. These algorithms are based on the principle of selecting the 'most constrained variable'; the heuristics attempt to fail as soon as possible when instantiating variables, what leads to reinstantiate the variables with other values, and so eliminate search subregions of considerable size. These heuristics are the following:

**Brelaz.** This heuristic was designed for solving graph coloring problems. For a partial coloring, the *saturation* degree of a vertex is the number of different colors used to color the adjacent vertices. For our problem, the heuristic selects first the variable with maximum saturation degree (the variable with fewer values in

its domain). Then, it breaks ties by selecting the variable with maximum degree (the degree for a variable is the number of adjacent uninstantiated variables).

**Rho.** This heuristic selects first the variable that maximizes equation $\rho = \prod_{c \in C - C_i}(1 - p_c)$. That is, the variable that minimizes $\prod_{c \in C_i}(1 - p_c)$, where $C$ is the set of constraints in the problem, $C_i$ is the set of incident constraints in the current variable $V_i$ and if a constraint $c$ in average limits a fraction $p_c$ of possible assignments, a fraction $1 - p_c$ is allowed. Thus this heuristic selects first the variable with the most and/or tightest constraints. The idea behind it is that by selecting this variable, the remaining subproblem contains a larger number of solutions (solution density $\rho$). The heuristic, however, does not take into account the available domain of the variables.

**Kappa.** This heuristic selects the variable in such a way that the parameter $\kappa$ is minimized, where $\kappa$ is a measure over the subproblem left after extracting the variable $V_i$ and is given by the following equation: $\kappa = \frac{-\sum_{c \in C} \log_2(1-p_c)}{\sum_{v \in V} \log_2(d_v)}$ where $V$ is the set of variables in the problem, and $d_v$ is the domain size of variable $v$. This heuristic depends on the proposal by Gent et al. [11] in which $\kappa$ captures the notion of the constrainedness of an ensemble of problems. The problems with $\kappa \ll 1$ are likely to be under-constrained, and solvable, whereas if $\kappa \gg 1$, these problems are likely to be over-constrained and unsolvable. Similarly as the heuristic Rho, this heuristic intends to select a variable that will leave a subproblem with high probability of being solvable.

More formal description of each heuristic can be found in the work by Gent et al. [12].

### 2.1 Solution Approach

The method for solving the CSPs in this work is the Forward Checking (FC) algorithm. FC takes a variable from the uninstantiated ones, sets a value for it, and propagates constraints (it keeps consistency in the domains of the adjacent and remaining variables). If one of those variables finishes with an empty domain, then the algorithm chronologically backtracks (BT), otherwise it continues with the next variable. This algorithm was chosen because it provides updated information in relation to the unsolved subproblem in each iteration. This information is in fact used by the heuristics to determine the next variable to instantiate. In our hybrid approach, the FC algorithm invokes the GA which runs for a number of cycles and determines the variable (it may be one or more) to be instantiated. Figure 1 illustrates the implementation diagram.

The variable(s) to be instantiated by the FC algorithm are taken from the best individual in the last cycle of the GA, every time this is called. The selected variables are those placed to the left-most part of the chromosome (a permutation-based representation is used) where each gene represents the index of each uninstantiated variable. $n$ is the number of variables in the chromosome.

When the FC algorithm starts solving a given instance of CSP, it calls the GA, which initializes the population (popsize is $15n$) with randomly generated

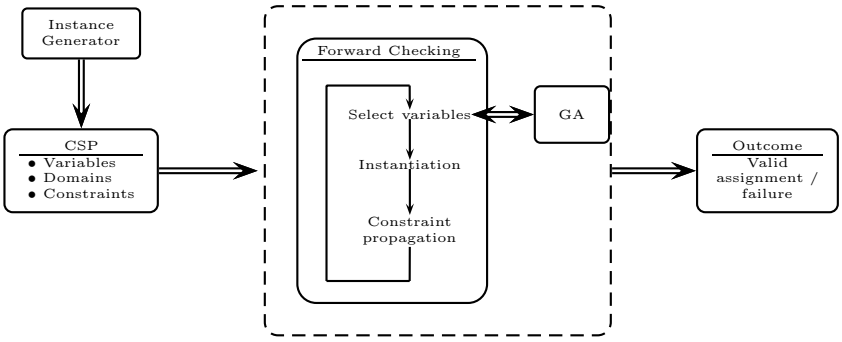



**Figure 1.** Flowchart of the proposed approach.

**Table 1.** Control parameters for the GA.

| Parameter | Initial | Following |
|---|---|---|
| Population | $15n$ | $10n$ |
| Cycles | $12n$ | $8n$ |
| Replacement | 60% | 60% |
| Crossover Probability | 90% | 90% |
| Mutation Probability | 10% | 10% |

chromosomes, runs for $12n$ cycles, and returns the selected variables for instantiation (*step*). FC is then used to assign values to *step* variables (the ones on the left), so there will be $n - step$ left to assign. For the subsequent invocations, the population in the GA is initialized based on the best chromosome of the previous call (we call it base chromosome and it is the one used to select the *step* variables). Now, the new population of size $10(n - step)$ is created, which is run for $8(n - step)$ cycles. The chromosome used to generate the new population is modified using random alterations. A copy of the base chromosome is also inserted in the population. The process continues until the complete CSP has been solved. The type of the proposed GA is steady state, with tournament selection, PMX crossover, and swap mutation. The GA was empirically tunned in its parameters and the final parameter set is presented in Table 1.

The objective function in the GA is given by the following expression:
$Ev = S_1 + nS_2$ where $S_1 = \sum_{i=1}^{step} \frac{T_i}{A_i(D_i)^2} {D_{max}}^2 (n-i)^2$ and
$S_2 = \sum_{j=step+1}^{n} D_j \left(\frac{j}{n}\right)^2$ being $n$ the number of variables remaining to be instantiated, *step* the number of variables the GA returns to the FC algorithm to be instantiated, $D_i$ the size of the current available domain for variable $V_i$, $D_{max}$ is the largest domain associated to a variable, $A_i$ is the number of adjacent variables to variable $V_i$ and $T_i = \sum_{j=step+1}^{n} \frac{\text{conf}_{i,j}}{D_i D_j}$ where $\text{conf}_{i,j}$ is the number of pairs in conflict between the current available values for variables $V_i$ and $V_j$.

The best individual is the one that maximizes the objective function above. This fitness function combines ideas from both the Brelaz and Kappa heuristics, specifically, with $S_1$ we are looking for those variables with small available domain and at the same time with constraints with high degree (with $\frac{T_i}{A_i}$), while

$S_2$ is used to emphasize that variables with small available domain should be selected first (shifted to the left side). From this expression, it can be observed that for smaller domains of the involved variables the value on $S_1$ would increase. It is also beneficial to have those variables to the left of the chromosome, and this is achieved by introducing the factor $(n-i)^2$. The value that makes a difference between two or more variables with the same minimal domain is $T_i$ which is a sum of the tightness for each uninstantiated variable and adjacent to variable $V_i$. The quotient of $T_i$ divided by $A_i$ in $S_1$ would give us an idea of how 'hard' in average are the constraints linking $V_i$ with the rest of the variables, considering only their available values. $S_2$ gives preference to select first those variables with smaller domain. This effect is achieved by maximizing the sum of the available domains of the remaining variables. Taking advantage on this, we also consider shifting variables with smaller available domains to the first positions in the chromosome. We give a weight to each position with $(j/n)^2$. It is also important to stress that when the FC algorithm is combined with heuristics Rho, Kappa, or Bz, just a single variable is returned for instantiation, that is, parameter *step* is only applies for the FC-GA combination.

## 3 Experiments and Results

This section presents the most important results obtained by the proposed approach. The experiments are divided following two different ways for generating the instances: in the first one, the generated problem instances have the parameter $p_2$ constant, that is, all constraints have the same number of inconsistent pairs; while in the second one, this parameter is randomly varied, leaving different number of inconsistent pairs between constrained variables. This way of generating the instances allows to observe the behavior of the various heuristics when the parameters used to define an instance are not uniform. For both cases, the performance of the algorithms is based on the number of consistency checks performed by the FC algorithm, with the aim to minimize it. The number of consistency checks is the most common way of comparing algorithms of this kind when solving constraint satisfaction problems, but there exist two other usual criteria such as the number of expanded nodes in the search tree and the number of backtracks. In the case of the GA, results report the average and best result over ten runs of the same instance.

**Problem instances with uniform $p_2$**
We present results for instances with 10 and 20 variables and domain size of 10. For both sizes, three different experiments are carried out, each one with a different value for parameter $p_1$ (constraint density).

First, we report results for instances with 10 variables. 40 random instances were generated for each value of $p_2$, the FC algorithm is executed with each heuristic and each instance, and then the average number of consistency checks is computed. That is the number plotted in the figures. For the GA case, each instance is run 10 times and then their average is used to obtain the average over the 40 instances, which is the one reported in the figures. $p_2$ is increased

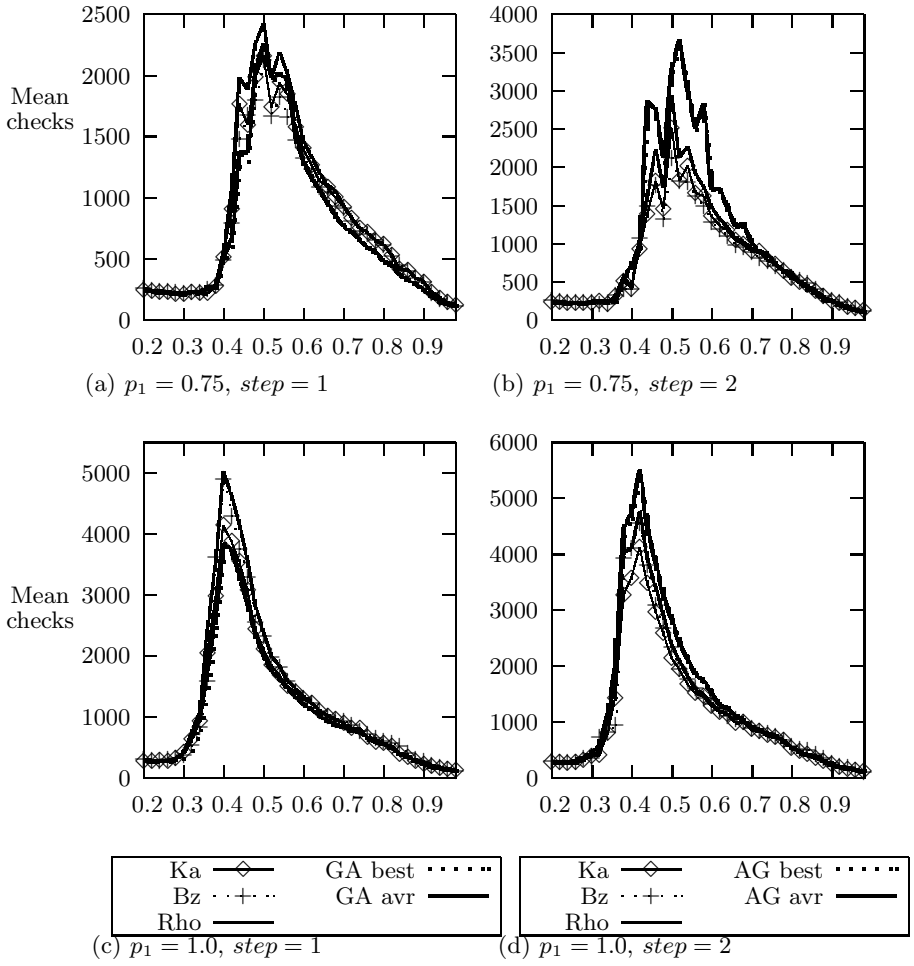



**Figure 2.** Result on problems $\langle 10, 10 \rangle$ with uniform $p_2$.

from 0.2 to 0.98 with steps of 0.02, trying to observe in detail the behavior of each heuristic in this range, and specifically over the phase transition (the region where the most difficult instances can be found). Figure 2 shows the first three series of experiments. In Figures 2 ((a) and (c)) the value for $p_1$ is 0.75 and 1, respectively.

The parameter *step* is set to 1. It can be observed that for all values of $p_1$, the GA approach shows slightly better results than the other single heuristics. It is also shown that there is an improvement from the approach when the instances have higher constraint density, for example, when $p_1$=1, it is clear that the FC-GA combination achieves better results (see Figure 2 (c)). Parameter *step* was intended to allow the FC-GA combination to select more than one variable to

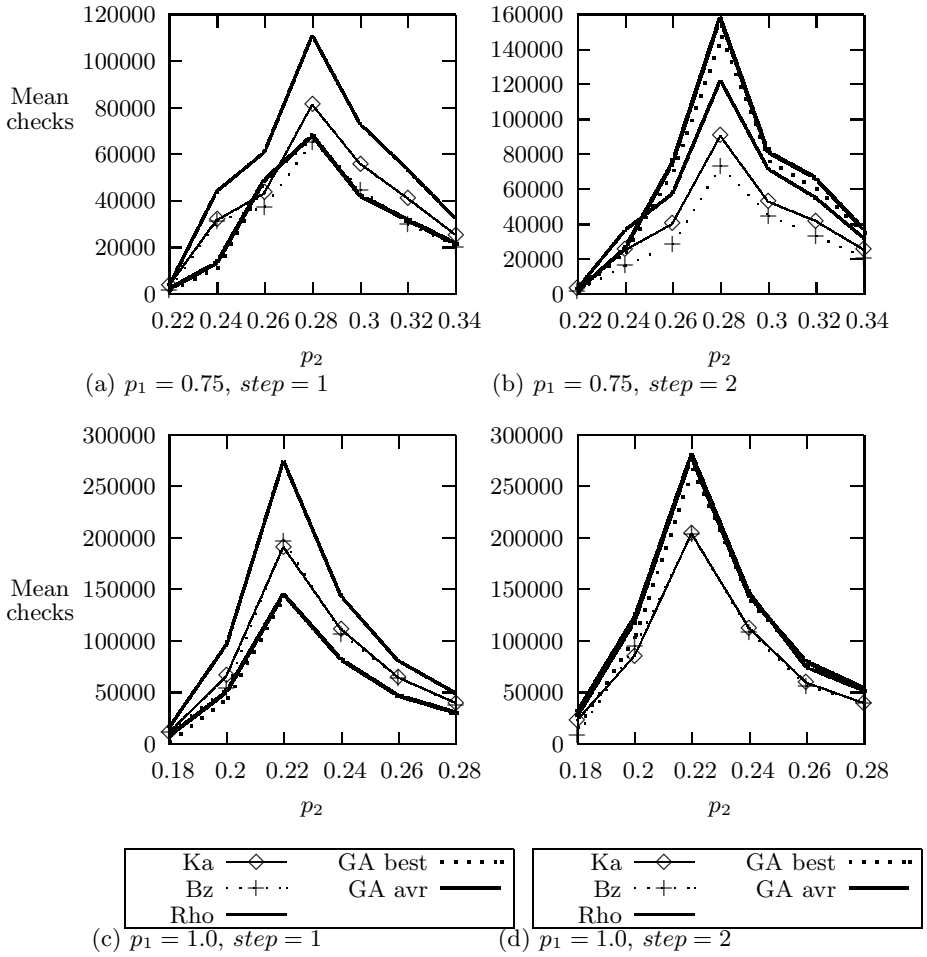



**Figure 3.** Results on problems $\langle 20, 10 \rangle$ with uniform $p_2$.

instantiate at each invocation. When we set $step = 2$ results can be observed in Figure 2 ((b) and (d)). We found for these cases and even when step had a higher value, single heuristics outperform our strategy.

The connection we deduced from these results is that indeed assigning a value to the most-left variable produces changes in the domains of the uninstantiated variables including that one selected by the parameter *step*. Consequently, those changes affect the domains of this variable, and so its selection is no longer the best one.

We now present results for instances with 20 variables. In this case, 20 different instances were randomly generated and tested with the FC algorithm and for each different value of $p_2$. The average number of consistency checks is reported in the figures. For the GA, the figures report the average of the average

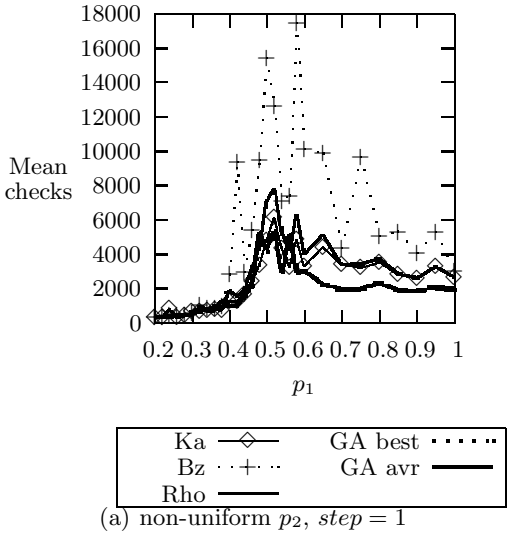
of ten runs for each instance, the same way as in the previous experiment. $p_2$ is increased by steps of 0.02. For these results, we concentrate in the range of values for $p_2$ between 0.22 and 0.34, where the phase transition appears. See Figure 3. It is clear in Figures (a) and (c) that the GA performs fewer number of constraint checks when *step*=1. Again, the GA shows better performance for highly constrained problems ($p_1 = 0.75$ and $p_1 = 1$). However, it is also true that as we increase the value in *step* to 2 (see Figure 3 (b) and (d)), the the advantage of the GA with respect to the other heuristics is less evident. In fact, we ran experiments for greater values of *step* (up to 5), but the best performance was found when *step*=1.

**Problem instances with non-uniform $p_2$**
In this set of experiments, the probability for inconsistent values between constrained pairs of variables ($p_2$) is not uniform. Because of this non-uniformity, a given variable may have more information, in addition to its available domain and degree, to be considered when selecting variables for instantiation. Heuristics Rho and Kappa, as well as our approach, exploit this situation. It is not the case with heuristic Bz, so that it is expected to produce different behavior in the results for the various heuristics.

For these experiments, instances have 20 variables, domain size of 10, and other additional particular features were considered. Specifically, for 15% of the constraints in an instance, parameter $p_2$ was set to 0.8, while for the remaining 85% of constraints, the same parameter has a value of 0.2. Now, what is interesting to observe is the behavior of the heuristics when the constraint density is varied ($p_1$). This parameter varies from 0.2 through 1 with steps of 0.02. For each value of $p_1$, 20 random instances were generated, each one was run 10 times, and the average number of consistency checks was computed.

Figure 4 (a) shows results when comparing all heuristics and the GA approach with *step*=1. As expected, the performance of heuristic Bz is very poor with respect to the other heuristics. The GA clearly beats all other heuristics for a wide range of values for $p_1$, including in those regions where the FC algorithm has its largest computational effort in combination with any heuristic. It is always possible, however, that by using either Kappa or Rho, a better result can be obtained for a particular instance, but let us recall that the result reported here in the GA case, is an average over a set of instances for each value of $p_1$. When observing results on experiments for *step*=2, Kappa, in general, has better performance than Rho and the GA. Nevertheless, the GA presents a reasonable performance over these instances, caused by the inclusion of the constraint density in the fitness function.

In order to support our study, statistical tests were run to validate the results. Despite of this, one may wonder about the overall performance of our strategy given that the computational cost of the GA is naturally higher given his population-based approach. It is then interesting to explore the trade-off between the gain in the number of constraint checks produced by the FC algorithm against the computational cost by any of the heuristics used including our approach. Results confirm the outcome on the previous experimentation. For




(a) non-uniform $p_2$, $step = 1$

**Figure 4.** Results on problems $\langle 20, 10\rangle$ with non-uniform $p_2$.

instance for less dense graphs with $p_1 = 0.5$ the best heuristic is Bz, and the GA has to work 140% more. But for dense graphs with $p_1 = 1$ the GA clearly generates better results on the number of constraint checks, with an additional cost of only 34% with respect to the effort taken by the Kappa heuristic. We think that with additional refinement of the FC-GA combination this percentage can be reduced, but this work is contemplated in future extensions of this investigation.

## 4 Conclusions

This article has proposed an innovative approach for using a GA to generate a dynamic variable ordering when solving CSPs. Using this scheme, under certain configuration of the GA, results are efficient, in terms of consistency checks. After testing for different values of parameter *step* (the number of variables to be instantiated before calling the GA again), it was found that best performance is shown when *step*=1. By establishing *step*=1, the fitness function in the GA could be seen as a deterministic heuristic that can be used to evaluate each of the uninstantiated variables and select that variable which maximizes the measure. This can be used as a single heuristic without considering the GA and probably would obtain as good results or better than those provided by the GA.

The FC-GA combination shows in general better results than the other heuristics, especially for highly constrained problems. It was also observed that when the probability $p_2$ is not uniform, the GA has a very competitive performance, achieving in some cases much better results than the other heuristics. The

success of the GA in these cases is that the fitness function takes into account the degree of a variable.

## Acknowledgments

This research was supported by ITESM under the Research Chair CAT-010 and the CONACyT Project under grant 41515.

## References

1. E. Tsang. *Foundations of Constraint Satisfaction.* Computation in Cognitive Science. Academic Press, London, 1993.
2. V. Kumar. Algorithms for constraint satisfaction problems: A survey. *AI Magazine 13(1):32-44*, 1992.
3. R. Dechter and I. Meiri. Experimental evaluation of preprocessing algorithms for constraint satisfaction problems. *Artificial Intelligence*, 68(2):211-242, 1994.
4. P.W. Purdom. Search rearrangement backtracking and polynomial average time. *Artificial Intelligence*, 21:117-133, 1983.
5. D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Alabama, 1989.
6. B. G. W. Craenen, A.E. Eiben, and J.I. van Hemert. Comparing evolutionary algorithms for binary constraint satisfaction problems. *IEEE Transactions on Evolutionary Computation*, 7(5):424–444, 2003.
7. A. E. Eiben. Evolutionary algorithms and constraint satisfaction: Definitions, survey, methodology and research directions. pages 13–58, 2001.
8. P. Prosser. Binary constraint satisfaction problems: Some are harder than others. In *Proceedings of the European Conference in Artificial Intelligence*, pages 95–99, Amsterdam, Holland, 1994.
9. B.M. Smith. Phase transition and the mushy region in constraint satisfaction problems. In *Proceedings of the European conference in Artificial Intelligence*, pages 100–104, Amsterdam, Holland, 1994.
10. B. M. Smith. Constructing an asymptotic phase transition in binary constraint satisfaction problems. *Journal of Theoretical Computer Science (Issue on NP-Hardness and Phase Transitions)*, 265(1):265–283, 2001.
11. I.P. Gent, E. MacIntyre, P. Prosser, and T. Walsh. The constrainedness of search. In American Association for Artificial Intelligence, editor, *In Proceedings af AAAI-96*, 1996.
12. I.P. Gent, E. MacIntyre, P. Prosser, B.M. Smith, and T. Walsh. An empirical study of dynamic variable ordering heuristics for the constraint satisfaction problem. In *Proceedings of CP-96*, pages 179–193. Springer, 1996.

# Two Hybrid Tabu Scatter Search Meta-heuristics for Solving MAX-SAT Problems

Dalila Boughaci[1], Habiba Drias[2] and Belaid Benhamou[3]

[1] University of Sciences and Technology – ITS- USTHB
BP 32, El-Alia Beb-Ezzouar, 16111, Algiers, Algeria
Dalila_info@yahoo.fr
[2] Institute of Computer Science –INI-
BP 68M OUED SMAR EL HARRACH , ALGER
drias@wissal.dz
[3] LSIS - UMR CNRS 6168 MARSEILLE CEDEX 20
belaid.benhamou@cmi.univ-mrs.fr

**Abstract.** Tabu search is a meta-heuristic that has been successfully applied to hard optimization problems. In this paper, two new hybrid meta-heuristics are studied for the NP-Complete satisfiability problems, in particular for its optimization version namely MAX-SAT. At first, we present a tabu scatter search approach, TS+SS, which is a tabu search procedure extended by a commonly shared collection of scatter elite solutions. Then, we introduce a scatter tabu search approach, SS+TS, which is a scatter search procedure enhanced with a tabu search improvement strategy. Experiments comparing the two approaches for MAX-SAT are presented. The empirical tests are performed on DIMACS benchmark.

## 1 Introduction

Tabu search is one of the meta-heuristic methods. It has been applied to various optimization problems with a great success. In this work, we propose two hybrid approaches based on tabu search meta-heuristic to solve the satisfiability problems. Given a collection $C$ of $m$ clauses involving $n$ Boolean variables, the satisfiability problem is to determine whether or not there exists a truth assignment for $C$ that satisfies the $m$ clauses. A clause is a disjunction of literals. A literal is a variable or its negation. A formula in conjunctive normal form (CNF) is a conjunction of clauses. The formula is said to be satisfiable if there exists an assignment that satisfies all the clauses and unsatisfiable otherwise. In the latter situation, we are interested in other variants of SAT. We mention among them the maximum satisfiability problem (MAX-SAT). The latter consists in finding an assignment that satisfies the maximum number of clauses. MAX-SAT is an optimization variant of SAT. They are an important and widely studied combinatorial optimization problem with applications in artificial intelligence and other areas of computing science. The decision variants of both SAT and MAX-SAT problems are NP-Complete [4, 8].
Many algorithms have been proposed and important progress has been achieved. These algorithms can be divided into two main classes:

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 45-54*

- *Complete algorithms*: dedicated to solve the decision version of SAT problem. The well- known algorithms are based on the Davis-Putnam-Loveland procedure [5]. Satz [13] is a famous example of a complete algorithm.
- *Incomplete algorithms*: they are mainly based on local search and evolutionary algorithms. Local search[16], tabu search [14, 1, 2], simulated annealing [10], genetic algorithms [7 ], GRASP[15], scatter search [6] and recently memetic algorithms [3] are examples of incomplete algorithms for SAT. These meta-heuristics are a good approach for finding a near solution of very large instances, in particular for unsatisfiable or unknown instances.

In this paper, we propose, at first, a tabu scatter hybrid procedure for MAX-SAT problems. Its algorithmic backbone is a tabu search (TS) which is extended by a commonly shared collection of elite solutions. This collection is maintained by the tabu search, which inputs quality solutions and is used by the scatter search to construct combined solutions. Then, a scatter search variant is proposed for the same problem. Its algorithmic backbone is a scatter search (SS) combined with a tabu search (TS) improvement strategy. The latter performs an intensified search of solutions around the scatter search regions. Experiments comparing the two approaches for MAX-SAT are presented. The empirical tests are performed on some well-known DIMACS benchmark instances. The paper starts with a brief review of the tabu search. Section 3 introduces the scatter search approach. Section 4 presents our new tabu scatter search approach. Section 5 presents the scatter tabu search approach. Our comparative study and experiments results are summarized in section 6. Finally, conclusion and future work are explained in section 7.

## 2 A Tabu Search Meta-heuristic

Tabu search is a meta-heuristic that has been proposed by Fred Glover [9]. It has been applied to various optimization problems including the satisfiability problem [14, 1, 2] and job shop scheduling [17]. Tabu search starts with an initial configuration generated randomly, then, the best neighbor solutions are selected. Tabu search uses also a list called "tabu list" to keep information about solutions recently selected in order to escape the solutions already visited. In the case where a tabu move applied to a current solution gives a better solution; we accept this move in spite of its tabu status by aspiration criterion. The search stops when the quality of the solution is not improved during a maximum number of iterations or when we reach a global optimal.

### 2.1 Tabu Search Items

In order to use tabu search for solving MAX-SAT problem, we define the following items:

- **A Solution** is represented by a binary chain X (*n* Vector); each of whose components xi receives the value 0 or 1. It is defined as a possible configuration verifying the problem constraints and satisfying the goal that consists in finding an assignment of truth values to the *n* variables that maximizes the sum of satisfied clauses.

**- A move** is an operator used to generate neighbor solutions. An elementary move consists in flipping one of the variables of the solution. The neighborhood of a solution is constituted by all the solutions obtained by applying an elementary move on this solution. A variable is in tabu state if it has been modified during the current move and it keeps it during a certain number of iterations called **tabu tenure**.
**- A Tabu List** is used to keep information about the solutions already visited in order to escape local optima by searching in new regions not already explored.

## 3 A Scatter Search Meta-heuristic

Scatter search [12] is a population-based meta-heuristic. It is an evolutionary method that constructs solutions by combining others**.** The approach starts with an initial population (collection of solutions) generated using both diversification and improvement strategies, then, a set of best solutions (reference set that incorporates both diverse and high quality solutions) are selected from the population. These collections of solutions are a basis for creating new solutions consisting of structured combinations of subsets of the current reference solutions.

### 3.1 A Scatter Search Template

Four methods are used to achieve the scatter search template:

**- A Diversification Generator.** The generator creates, from a seed solution, a collection of diverse solutions, applies a heuristic process for improving these solutions and designates a subset of the best solutions to be reference solutions. Solutions gain membership to the reference set according to their quality or their diversity.

**- An Improvement Method.** An Improvement method transforms a trial solution into one or more enhanced trial solutions. To improve the quality of solutions we often apply a heuristic process.

**- A Subset Generation Method.** A subset generation method operates on the reference set (collection of elite solutions), to produce a subset of its solutions as a basis for creating a combined solution.

**- A Combination Operator.** A solution combination method transforms a given subset of solutions created by the subset generation method into one or more combined solutions. In this step, we create new points consisting of structured combinations of subsets of the current reference solutions.

## 4 A Tabu Scatter Search

In order to take advantage of the individual benefits of a single-solution oriented approach and a population oriented approach, we propose a hybrid tabu scatter search

approach. Its backbone is a basic tabu search that works on a single solution by building neighborhoods from which a best admissible candidate is passed to the scatter search process. The hybrid tabu search makes use of a population based strategy and maintains a collection of elite solutions. More precisely, the tabu scatter hybrid (TS+SS) procedure starts with an initial solution generated randomly; then, a basic tabu search is started. The duration of this second phase (TS) is given by an input parameter $iter_{TS}$ corresponding to the number of iterations of the basic TS process ( see code below). During the TS phase a new best solution is always deposited into the collection. Every "$iter_{SS}$" iterations the algorithm calls the subroutine of the scatter search phase, operating on the solutions in the collection. Those solutions represent the reference set in the basic scatter search. They are a basis for creating new combined solutions using a combination operator. The combination method, that we have used, randomly selects a position *K* to be the crossing point from the range [1,…n]. The first *K* elements are copied from the one reference point while the second part is copied from the second reference point to create the new trial solution. After having built new combined solutions via the combination method cited above, the best solution is returned to TS to serve as an initial starting point which may be enhanced after resetting the tabu list. The algorithm terminates after a certain number of iterations.

### 4.1 A Tabu Scatter Search Outline

```
Step 1. Initialization
  Set tabu scatter search(TS+SS)parameters
  //iter is the current iteration of TS+SS process,
  //maxiter is the maximum number of iterations of TS+SS
  //iter1 is the current iteration of TS process,
  // iter_TS is the maximum number of iterations of TS, //iter_Ss
  is the number of iteration in which the scatter search
  (SS )is called,                                    // TL
  is the tabu list,
  // S* is the best solution with the minimum F* corresponds
  to S*, F* objective function value that is F*=F(S*),
  - Generate an arbitrary solution S;                        -
  Evaluate F (S); S*= S; F* = F; iter=0; iter1 = 0;
Step 2. Iteration
 While (iter < maxiter) do
  begin
    While (iter1 < iter_TS  and iter < maxiter)  do
      begin
      - iter  = iter + 1; iter1 = iter1 +1;
       - Apply a basic TS process;/* iteratively execute
        iter_TS iterations using neighborhood operators*/
       - Add the good solution found to the collection
         of elite solutions to construct the reference
         set for the next phase;
   If  (iter = iter_SS )  then
     begin
      /*while performing the TS, execute an SS
                phase every iter_SS */
         - Apply a TS process using the second move (SWAP) to
             diversify the search, and add the diverse
             solution to the collection;
```

```
          - Generate subsets of the reference set as a basis
              for creating combined solutions;
          - For each subset produced, use the combination
              operator to produce new solutions;
          - Improve the combined solutions
       end;
 end;
 - iter1=0;
end;
end;
Step 3. Termination. Print the best solution with the best
objective value.
```

## 5 A Scatter Tabu Search

Our scatter tabu search-based evolutionary approach starts with an initial population of solutions created using a diversification generator. The latter creates, from a seed solution *V*, a collection of solutions associated with an integer *h (1<h<=n)*. A solution is represented by a binary chain *V* (*n* Vector). Two types of solutions *V'* and *V''* are created from the seed solution V and given as:

Type1 solutions $V'[1+k*h]= 1- V[1+k*h], \quad k=1,2,\ldots..n/h, k<n$.

Type2 solutions: *V''* are the complement of *V'*.

Then, each solution in the population makes tabu search to improve its fitness. After that, a set of solutions (reference set) are selected from the current population. The resulting reference set has *B1* high quality of individuals plus *B2* diverse solutions. The reference set is a basis for creating new solutions consisting of structured combination of subsets of the current reference set. The combination method (the same described in the preview section) is applied to all subsets of solutions of the current reference set. After having built new combined solutions, the combined solution is returned to TS procedure to serve as an initial starting point which may be enhanced. With all this components: diversification generator, reference set selection, combination method and intensified tabu search procedure, we hope to be able to achieve a good compromise between intensification and diversification in the search process. The search terminates after a certain number of generations or when we reach the optimum global.

### 5.2 A Scatter Tabu Search Outline

```
Step 1. Initialization
Set scatter tabu search parameters
//Psize is the size of the population P,
//B is the size of the reference set
//maxiter is the maximum number of generations,
// iter is the current generation,
- Call the Diversification generator to create an
  initial population P;
- Use TS to create enhanced trial solutions of P;
- Evaluate and order the solutions in P according
     to their objective function value;
- iter=0 ;
```

```
Step 2. Iteration
 While (iter < maxiter)do
  begin
      - Create the Reference Set of selections by
       choosing B1 high quality and B2 diverse
       solutions from P where B1+B2= B;
      - Subsets Generation Method
     While(continuing to maintain and update
         reference Set) do
      begin
        - For each subset produced, use the
         combination method to produce new solutions;
        - Use TS to create enhanced trial solutions;
         - Update the Reference Set: If the resulting
            solution improves the quality then add it to
            the B1 high quality solutions and remove the
            worst one else add it to the B2 solutions and
            removes the low diverse one in B2;
       end;
         - Build a new population P by initializing the
             generation process with the reference set;
        - Use TS to create enhanced new trial solutions;
        - iter= iter+1;
     end;
Step 3. Termination, print the best solution with the best
objective value.
```

## 6 A Comparative Study

The purpose of this comparative experiment is to evaluate the performance of each one of the proposed techniques to solve MAX-SAT instances. First of all, we compare on the table 1 the approaches regarding their principles and the operators used by each approach. Further, we give some numerical results obtained by applying each algorithm on MAX-SAT instances. The objective is to explore the influence of population and combination strategies by comparing SS+TS and TS+SS. To compare the hybrids and to explore the influence of the hybridization, we have compared SS+TS and TS+SS with SS and TS alone. The results are given on the tables below.

### 6.1 Computational Results

All experiments were run on a 350 MHZ Pentium II with 128 MB RAM. All instances have been taken from the SATLIB [11]. They are hard Benchmark Problems. On each instance the different algorithms have been executed in order to compute the average of the maximum number of the sum of the satisfied clauses.

**The DIMACS Benchmarks**
Two kinds of experimental tests have been undertaken. The goal of the first ones is the setting of the different parameters of the TS+SS, and SS+TS algorithms like the Tabu tenure, the number of iterations, the population size and the interaction between the two algorithms parameters. These parameters are fixed as:

**Table 1.** Comparison of TS, SS, TS+SS, and SS+TS approaches

| | **TS** | **SS** | **TS+SS** | **SS+TS** |
|---|---|---|---|---|
| **Principles** | -neighbor search | - Evolutionary meta-heuristic | - neighbor search meta-heuristic | -Evolutionary meta-heuristic |
| | -Single current solution | - Population-based | -Population-based | - Population-based |
| | -Interdiction mechanism | Biological evolution | Interdiction mechanism | -Biological evolution and interdiction |
| **Operators** | -Move<br>-Tabu list<br>- Aspiration criterion | -Reference set selection<br>- Structured combination<br>- Improvement local technique | - Move<br>- Tabu list<br>-Aspiration criterion | - Reference set selection<br>- Structured combination<br>- Improvement local technique |
| **Solution or Population generation** | At Random | Using diversification generator | At Random or using a heuristic | Using a diversification generator |

**- TS+SS.** The maximum total number of iterations was set to maxiter=1000. The basic TS phase parameter, $iter_{TS,}$ was set to 100 iterations, the population size was set to 40, the SS procedure was called every $iter_{SS}$ = 10 iterations. The move operator for TS intensification phase was the variable flipping. A second move operator is used in order to diversify the search consisting in permuting between two variables chosen at random. This phase is executed before calling the SS subroutine, that, in order to create a collection of best solutions including diverse and high quality solutions.
**- TS.** The maximum total number of iterations was set to maxiter=10000. The move operator was the variable flipping. and tabu tenure was set to 7.
**- SS+TS**. The basic SS phase parameter, the maximum total number of iterations was set to maxiter=3, the reference set was set to 10, the population size was 100, and the TS parameter was set to 30 iterations and tabu tenure was set to 7.
**- SS**. is a scatter search with a simple local search as an improvement technique. The SS parameters are: the maximum total number of iterations was set to maxiter=3, the reference set was set to 10 and the population size was 100.
The second kind of experiments concerns MAX-SAT instances. All these instances are encoded in DIMACS CNF format [11]. The tables below show the results obtained by our algorithms. These columns contain the name of instance, the number of variables, the number of clauses, the solution found by each algorithm, and the algorithm running time in second. The results found are classed by class:

**AIM class**: Artificially generated random 3-SAT, defined by Kazuo Iwama, Eiji Miyano and Yuichi Asahiro [18]. We have chosen six instances.

**Table 2.** Solutions quality and running time results obtained by TS, TS+SS, SS and SS+TS on AIM instances.

| *Instance/ satisfiable* | # Var | # claus es | TS | TS Time | TS+ SS | TS+SS Time | SS | SS Time | SS+ TS | SS+ TS Time |
|---|---|---|---|---|---|---|---|---|---|---|
| *Aim50-1-1* | 50 | 80 | 78 | 21,6 | 79 | 44,8 | 79 | 9,6 | 79 | 54,8 |
| *Aim50-2-1* | 50 | 100 | 99 | 26,5 | 99 | 43,3 | 99 | 11,5 | 99 | 15,7 |
| *Aim50-3-1* | 50 | 170 | 169 | 44,3 | 170 | 12,0 | 167 | 18,9 | 165 | 25,5 |
| *Aim100--1* | 100 | 160 | 154 | 87,5 | 159 | 95,1 | 158 | 34,5 | 157 | 43.7 |
| *Aim100-2* | 100 | 200 | 195 | 105,5 | 199 | 156,1 | 196 | 46,8 | 195 | 63.6 |
| *Aim100-3* | 100 | 340 | 334 | 185,6 | 335 | 243.9 | 330 | 70,9 | 327 | 183.9 |

**JNH class**: Randomly generated instances- constant density model. The instances have originally been contributed by John Hooker [18].

**Table 3.** Solutions quality and running time results obtained by TS, TS+SS, SS and SS+TS on JNH instances.

| *Instance satisfiable* | # Var | # claus es | TS | TS Time | TS+ SS | TS+SS Time | SS | SS Time | SS+ TS | SS+TS Time |
|---|---|---|---|---|---|---|---|---|---|---|
| Jnh201- yes | 100 | 800 | 797 | 885,3 | 800 | 155,7 | 795 | 31.4 | 799 | 670.2 |
| Jnh202- no | 100 | 800 | 792 | 1084,2 | 796 | 896,0 | 795 | 30,6 | 797 | 777.3 |
| Jnh203-no | 100 | 800 | 798 | 1159,0 | 796 | 1135,1 | 790 | 31,3 | 794 | 753.7 |
| Jnh204-yes | 100 | 800 | 796 | 707,7 | 798 | 728,7 | 796 | 28,7 | 797 | 821.1 |
| Jnh205-yes | 100 | 800 | 800 | 697,3 | 799 | 894,1 | 794 | 34,5 | 797 | 802,2 |
| Jnh206-no | 100 | 800 | 799 | 701,1 | 797 | 750,2 | 794 | 30,4 | 796 | 821.1 |
| Jnh207-yes | 100 | 800 | 797 | 701,4 | 797 | 841,8 | 793 | 28,3 | 796 | 831,4 |
| Jnh208-no | 100 | 800 | 797 | 700,7 | 794 | 761,1 | 795 | 34,2 | 796 | 789.1 |
| Jnh209-yes | 100 | 800 | 797 | 697,8 | 797 | 797,8 | 794 | 27,8 | 796 | 748.0 |
| Jnh210-yes | 100 | 800 | 800 | 699,0 | 800 | 83,2 | 795 | 28,6 | 797 | 767.6 |

**Parity8 class**: Instance arises from the problem of learning the parity function. Defined by James Crawford (jc@research.att.com). All the instances are satisfiable by construction [18].

The results obtained by the different approaches are acceptable (we have reached the optimum for some instances in reasonable time). In many cases (Jnh 201, Jnh 210, par8-1-c for example), the tabu scatter search (TS+SS) performs the other approaches in solving such instances. In some case ( Jnh 205, for example) a TS alone performs the others. Also, for some benchmarks, the SS alone performs the SS+TS which means that the choice of adequate parameters for a meta-heuristic in solving a given

benchmark is a difficult operation and the hybridization in some situation is not very interesting. However, for the most benchmarks the hybridization improve the quality of solutions and gives a good result. So, according to our results, we can see that, in general, when SS is incorporated in TS (TS+SS), the solutions space is better searched. When intensified improvement tabu search and diversified components are incorporated in SS (SS+TS), the solutions space is better searched but the process search takes more time to find a solution. We precise, that the role of a tabu search technique in scatter search is to locate the solution more efficiently.

**Table 4.** Solutions quality and running time results obtained by TS, TS+SS, SS and SS+TS on parity8 instances.

| *Instance* | # Var | # clauses | TS | TS Time | TS+ SS | TS+SS Time | SS | SS Time | SS+ TS | SS+TS Time |
|---|---|---|---|---|---|---|---|---|---|---|
| Par8-1 | 350 | 1149 | 1115 | 1768,1 | 1126 | 986,7 | 1141 | 76,7 | 1141 | 303,94 |
| Par8-1-c | 64 | 254 | 250 | 68,9 | 254 | 10,9 | 248 | 4,1 | 245 | 121.12 |
| Par8-2 | 350 | 1157 | 1114 | 1813,6 | 1137 | 937,9 | 1146 | 76,0 | 1146 | 401,99 |
| Par8-2-c | 68 | 270 | 267 | 78,9 | 267 | 137,7 | 263 | 5,13 | 260 | 99,17 |
| Par8-3 | 350 | 1171 | 1127 | 1850,6 | 1154 | 1131,0 | 1162 | 72,8 | 1162 | 536.33 |
| Par8-3-c | 75 | 298 | 296 | 96,2 | 294 | 971,4 | 291 | 11,8 | 289 | 117.23 |
| Par8-4 | 350 | 1155 | 1105 | 1816,9 | 1154 | 1351 | 1149 | 70,9 | 1149 | 278.26 |
| Par8-4-c | 67 | 266 | 261 | 74,5 | 264 | 141,2 | 260 | 4,8 | 260 | 91,06 |
| Par8-5 | 350 | 1171 | 1110 | 1842,7 | 1164 | 1581.1 | 1163 | 74,8 | 1163 | 584.16 |
| Par8-5-c | 75 | 298 | 293 | 95,9 | 294 | 157,0 | 290 | 7,2 | 290 | 119,91 |

## 7 Conclusion and Perspectives

In this paper, we have presented, at first, the single–oriented meta-heuristic called tabu search. We have proposed to hybridize it with a scatter search evolutionary algorithm.. Then, we have presented a scatter tabu search approach. The proposed approaches have been implemented for solving MAX-SAT hard instances. Our objective is to explore the influence of both population and combination strategies on the ability of generating high quality solutions in single solution-oriented approaches and vice versa. We have shown that the impact of a population strategy on a single-oriented approach is like the import of a local search in a population- based approach. After an intensified experimentation, we conclude that the tabu search (TS) can be considered as a powerful procedure capable to organize and to direct operations of subordinate methods. We plan to improve our framework by implementing a parallel environment including the two approaches.

## References


1. D.Boughaci H.Drias "Solving Weighted Max-Sat Optimization Problems Using a Taboo Scatter Search Meta-heuristic". *In Proceedings of ACM* Sac 2004, pp35-36, 2004.
2. D.Boughaci, H.Drias. "PTS: A Population-based Tabu Search for the Maximum Satisfiability Problems". *In Proceedings of the 2 IEEE GCC conferences*, pp 622-625, 2004.
3. D.Boughaci, Drias H and Benhamou B. "Solving Max-Sat Problems Using a memetic evolutionary Meta-heuristic". *In Proceedings of 2004 IEEE CIS*, pp 480-484, 2004.
4. S.Cook . "The Complexity of Theorem Proving Procedures". I*n Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing, 1971*.
5. M.Davis, G. Putnam and D.Loveland. "A machine program for theorem proving". C*ommunications of the ACM*, 394-397, Jul 1962.
6. H.Drias. "Scatter search with walk strategy for solving hard Max-W-Sat problems". I*n Proceedings of IEA- AIE2001. Lecture Notes in Artificial Intelligence, LNAI-2070*, Springer-Verlag, Boudapest, pp 35-44, 2001.
7. J.Frank, "A study of genetic algorithms to find approximate solutions to hard 3CNF problems". *In Proceedings of Golden West International Conference on Artificial Intelligence,1994.*
8. MR.Garey, S.Johnson. "Computer and intractability, a guide to the theory of NP-Completeness, Freeman company. Sanfranscsco, 1978.
9. F. Glover, "Future paths for integer programming and links to Artificial intelligence", *Operational Research*, Vol 31, 1986.
10. P.Hansen P; B.Jaumard. "Algorithms for the Maximum Satisfiability". Journal of Computingm 44- pp 279-303, 1990.
11. H. H. Hoos and T. Stutzle. SATLIB: An online resource for research on SAT. In I. Gent, H. van Maaren, and T. Walsh, editors, *SAT2000: Highlights of Satisfiability Research in the year 2000*, Frontiers in Artificial Intelligence and Applications, pp 283–292. Kluwer Academic, 2000.
12. M.Laguna, F.Glover "*Scatter Search"*, Graduate school of business, University of Colorado. Boulder, 1999.
13. CMLi and. Anbulagan, "Heuristic based on unit propagation for satisfiability". In Proceedings of CP 97, springer-Verlag, LNCS 1330, pp 342-356, Austria, 1997.
14. B.Mazure, L.Sais and E.Greroire,." A Tabu search for Sat". In Proceedings of AAAI 1997.
15. PM.Pardalos, L.Pitsoulis and MGC. Resende. "A Parallel GRASP for MAX-SAT Problems. PARA96 Workshop on Applied Parallel Computing in Industrial Problems and Optimization", Lynghy. Denmark August 18-21, 1996.
16. B.Selman , H. .Kautz and B.Cohen . "Local Search Strategies for Satisfiability Testing". P*resented at the second DIMACS Challenge on Cliques, Coloring, and Satisfiability, October 1993.*
17. Taillard , "Parallel Tabu Search Techniques for Job Shop Scheduling Problem". *ORSA Journal on Computing*, 6:108-117, 1994.
18. http://www.satlib.org.

# Complete Instantiation Strategy for Approximate Solution of TCSP

Priti Chandra, Arun K. Pujari

A I Lab, University of Hyderabad, India
priti_murali@yahoo.com

**Abstract.** Interval algebra(IA) based temporal constraint satisfaction problems (TCSPs) are useful in formulating diverse problems. The usual approach to solve IA networks is based on partial instantiation strategy - backtrack search for exact solution. To the best of our knowledge, determining approximate solution for TCSPs is not addressed so far. In this paper we propose a new complete instantiation strategy based on a complete algorithm to determine an approximate solution of IA networks. We identify a property of constraints called *nastiness* that disturbs monotonic nature of entropy of a constraint. We go beyond the identification of nasty constraints to pin-point the singleton to restore normal behaviour of entropy. On termination, the algorithm guarantees either an exact or an approximate solution depending upon the number of constraints the solution violates. We demonstrate experimentally that solution to general IA networks can be efficiently obtained in time polynomial in the size of the network with the success rate of 95% contrary to exponential exact algorithm.

Keywords: Constraint satisfaction problem, Approximation algorithm, Interval algebra

## 1 Introduction

Constraint Satisfaction Problems (CSP) are in general NP-hard class [6]. On the other hand, CSPs have numerous applications in almost all branches of engineering. There have been several attempts to devise solution techniques for CSPs. One approach is to characterize tractable subclasses and to provide polynomial-time algorithm for solving such instances. Another approach is to devise good heuristics and search strategies. In order to understand the distribution of hard instances, there have also been studies on identifying values of critical parameters which lie between the easy instances of under-constrained and of over-constrained instances. In this paper, we attempt to characterize the hardness of problem instances in a different manner. One wonders whether there are certain nasty constraints in an instance of CSP that is possibly the reason for hardness. And if so is the case, this paves the way to devise approximation scheme to solve hard instances by settling these nasty constraints. The efficiency of such an approximation method lies in settling very small number of nasty constraints to obtain a solution in polynomial time.

In this paper we study this aspect in the context of qualitative temporal CSP, namely Allen's framework [1] IA. We characterize a reason for late solution for prob-


*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 55-64*

lems due to presence of nasty constraints. The major contribution of this paper can be highlighted as follows.

In this paper, we introduce a new polynomial time complete method for determining an approximate solution of IA network. We start with a path consistent IA network. An iterative transitive closure algorithm such as weighted path consistency assigns highest weight to an atomic relation in a label that has maximum likelihood to be a feasible relation. Intuitively, when the highest weight relation on the edge agrees with the relation with highest weight in the constraint computed by averaging along all paths, this relation is the best possible candidate feasible relation in the constraint. A conflict occurs when the highest weight relation is not the same as that from the paths. This includes following two possibilities: (a) Highest weight relation on the edge is present in the averaged constraint, but with a lower weight, (b) Highest weight relation on the edge is not present at all in the averaged constraint. Following our intuition, in order to forcefully make the two agree, either we raise the lower weight of an existing relation to become the highest weight or we introduce a new atomic relation with highest weight. We term the constraint that exhibit the property of introducing new singletons as highest weight relation to resolve the conflict as *nasty constraint.* This adjustment of weights helps us to reduce the conflicts as and when they appear in an iteration. In case of conflicts, the solution may or may not violate any constraint. This helps in computing an approximate and early solution for hard instances. We prove that presence of nasty constraints in hard instances is responsible for preventing entropy of constraint from decreasing monotonically. Introducing the required singletons, restores the monotonic decrease in the subsequent iterations. Experiments reveal this method solves general IA networks by violation of small fraction of constraints.

In Section 2, we present IA framework and related work. In Section 3, we summarize weighted path consistency. In Section 4, we introduce the concept of entropy for weighted IA network with preliminary experiment. In Section 5, we propose that nasty constraints reflect the hardness of any given problem instance in IA with theoretical justification in Section 6, which contains the main result of this paper, a polynomial time complete algorithm for approximate solution of hard TCSP. We report experimental analysis in Section 7. Section 8 contains conclusions.

## 2 Interval Algebra and Related work

IA defines thirteen atomic relations that can hold between any two time intervals, namely before(b), meet(m), ovelap(o), start(s), during(d), finished-by(fi), equal(eq), finish(f), contain(di), started-by(si), overlapped-by(oi), meet-by(mi) and after(bi) [1]. In order to represent indefinite information, the relation between two intervals is a disjunction of the atomic relations. Reasoning for the complete interval algebra is known to be NP-hard [22]. Traditional solution techniques for temporal and spatial domains are either based on complete[8, 11, 18, 20] or partial instantiation strategies[21]. So far there is no complete method based on complete instantiation strategy for approximate solutions for qualitative TCSPs.

Research in phase transition is investigated [9, 7, 3, 5, 4] to study instance hardness. In the context of IA network, it is not possible to have any estimate of number of solutions. So far we roughly know the hard instances exist for a combination of parameters. The entropy based analysis of nasty constraints looks to be promising enough to open up a new study in this direction for qualitative CSPs. Basically to study a discrete problem consisting of only disjunctions, we are translating it to a continuous domain by adapting a weighted formalism.

## 3 Weighted path consistency

In this paper, we use weighted path consistency algorithm as proposed in [13], [2]. In a weighted IA network *W(N)* each constraint is represented as a 13-dimensional weight vector $W_{ij} \in R^{13}$ such that $0 \leq W_{ij}^{m} \leq 1$, $1 \leq m \leq 13$, $\Sigma W_{ij}^{m} = 1$. $W_{ij}^{m}$ denotes the weight of the atomic relation $IA_m$ in the constraint between variables *i* and *j*. The value 0 for $W_{ij}^{m}$ implies $IA_m$ is absent in the disjunction. We call each $W_{ij}$ as *weighted constraint*. Given an IA network *N*, we obtain the corresponding weighted network by assigning equal weights to all the atomic relations present in a constraint. We represent the IA-composition table [6] as a 3-dimensional binary matrix **M**, such that $\mathbf{M}_{ijm} = 1$ if and only if the atomic relation $IA_m$ belongs to the composition of the atomic relations $IA_i$ and $IA_j$. The composition of two weighted relations $W_{ik}$ and $W_{kj}$ resulting in a relation $W_{ij}(k)$ is denoted as $W_{ik} \otimes W_{kj}$. The intersection of two weighted relations $W_{ij}$ and $V_{ij}$ is denoted as $U_{ij} = W_{ij} \cap V_{ij}$, defined as follows [13]:

$$W_m^{ij}(k) = \frac{\sum_u \sum_v M_{uvm} W_u^{ik} W_v^{kj}}{\sum_m \sum_u \sum_v M_{uvm} W_u^{ik} W_v^{kj}}, \quad 1 \leq m \leq 13. \qquad U_m^{ij} = \frac{W_m^{ij} V_m^{ij}}{\sum_m W_m^{ij} V_m^{ij}}, \; 1 \leq m \leq 13$$

We follow a slightly different approach for computing the averaged constraint along the paths. For each edge, first we compute the non-zero normalized average of all vectors that are computed by path-wise composition. This averaged vector is intersected with the edge vector followed by normalization. We use intersection operator only once which reduces numerical computations. The weighted path consistency algorithm, unlike the conventional path consistency, modifies only the weights of constraints. The atomic relations with higher weights are more favorable to be the feasible ones, whereas those with smaller weights are less likely to participate in a solution. There will be no occasion when the weight values in the vectors will stop changing unless it is a network only with singleton labels(trivial case).

## 4 Entropy of IA network

In this section, we introduce the concept of entropy for IA network in the context of weighted formalism. In [14, 15, 16], the three properties of measures on entropy given in [19] are generalized. The Renyi's quadratic entropy (RQE) is given as $-\log \sum_m (W_m^{ij})^2$. In the context of minimizing entropy, for the sake of convenience,

*log* in the above expression is normally dropped. For the present study, we loosely define entropy to be without *log* [23].

**Definition 1**: Entropy of a weighted constraint $W^{ij}$ is defined as follows

$$E_w^{ij} = -\sum_m (W_m^{ij})^2 \quad where \quad 0 \le W_m^{ij} \le 1 \ \ and \ \ \sum W_m^{ij} = 1$$

**Definition 2**: Entropy of a weighted network is defined as follows

$$E_N = \sum_{ij} E_w^{ij}$$

The least entropy of a constraint corresponds to a singleton relation and the highest entropy is when it has non-atomic relations with equal weights. Path consistency algorithm indeed prevents the entropy of the constraint network from increasing.

**Theorem 1**: For a given network *N*, enforcing path consistency does not increase $E_N$ .

We illustrate this with the help of a simplex triangle (Figure 1) for a constraint with a maximum of three atomic relations. The three vertices **C**, **D** and **E** are the lowest entropy points that correspond to the three possible singleton labels for the constraint. The centre **A** corresponds to the highest entropy corresponding to equal weights for the three relations. The contours represent states with equal entropy. The entropy state at an edge indicates a conflict between the two singleton labels.

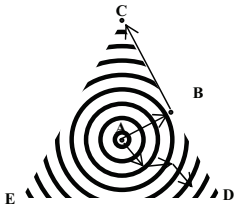



**Figure 1**. Simplex Triangle

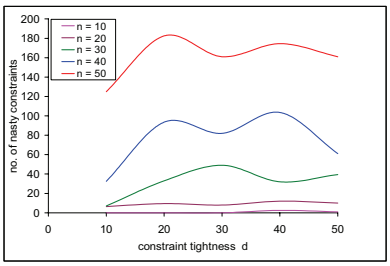



**Figure 2**. No. of constraints with fluctuating entropy for M(*n*,*d*,7) for known consistent problems.

In the conventional path consistency, we move from A to C in one step or from A to B and then possibly to C. In the event of a solution, the search ends at a vertex else it stops at either A or at B. Ideally, any search technique should choose a descending path from A to one of the vertices, say D (Figure 1). We have experimented initially with convex IA networks. For our experimental study, we generate random instances based on three parameters, namely network size(*n*), constraint tightness(*d*) and label size(*t*) [12], [13]. We experimented with 200 instances for each value of *n* in the range [10,100]. There was not a single instance out of randomly generated 3800 convex problem instances with any fluctuation in the entropy. We repeat the same experiment for general IA networks that are known to be consistent. We find that unlike the convex case, the entropy values of some constraints increase after an initial decrease, but again continue to decrease until stabilization i.e non-monotonic behaviour. Thus for convex network, the entropy value for every edge takes a descending path from the centre of the simplex to a boundary (Figure 1), a monotonic behaviour. On the other hand, the trajectory of entropy value of any edge in a non-convex network

need not be a descending path but a longer trajectory path. The search begins at the centre towards a vertex, but moves along the periphery of a contour with smaller entropy.

The number of constraints with fluctuations in entropy is very high for the problem instances in the hard region [12]. Figure 2 depicts the number of such constraints in instances with $t = 7$, $n$ in the range [10,50] and $d$ in [10, 50]. The peaks in the graphs (Figure 2) are the regions of hard instances (one to one correspondence of peaks is not possible due to the different model for generating problem instances used in this study). This has motivated us to study the behaviour of entropy in order to identify the cases when entropy increases. This study narrows down to basic fundamentals of multiplication of two vectors. The difference between entropy of a pair of vectors consisting of same number of non-zero entries, depends on the relative order of their highest value. When the number of non-zero entries in the two vectors is not same, then one cannot conclude clearly as which of the two will have a higher value of entropy. It depends on the relative distribution of the values within the vector. We formalize these observations as following results:

**Theorem 2:** Given two normalized vectors $U$ and $V$ with same no. of non-zero components, i.e. $nz(U) = nz(V)$, $|E(U)| \geq |E(V)|$ iff $\max(U) \geq \max(V)$, where $\max(U)$ is the component with highest value.

**Theorem 3:** Given two normalized vectors $U$ and $V$ such that $nz(V)>nz(U)$, then $\max(V)>\max(U)$ is not a sufficient condition for $|E(V)|<|E(U)|$.

## 5 Nasty Constraints

In this section, we identify a new property of weighted IA constraints called *nastiness* that is responsible for a difficulty in computing solution of a problem instance. Suppose $W_{ij}$ is the weighted constraint on the edge *(i,j)* and $W$ is the averaged constraint obtained from all possible paths using the weighted composition operator as explained in section 2. $W_{avg}$ is the constraint obtained by weighted intersection of $W_{ij}$ and $W$ at the end of the current iteration of weighted path consistency. We study the impact of replacing $W_{ij}$ by $W_{avg}$ in terms of weights of atomic relations that will increase or decrease with the help of inner product of vectors.

**Lemma 1**: Given two weighted IA constraints $U = [u_i]$ and $V = [v_i]$, the normalization factor $\lambda = \sum u_i v_i$ will satisfy the conditions, $u_{min} \leq \lambda \leq u_{max}$ and $v_{min} \leq \lambda \leq v_{max}$ where $u_i \in [u_{min}, u_{max}]$, $v_i \in [v_{min}, v_{max}]$.

**Theorem 4**: $W_{avg}[\text{argmax}(W_{ij})] > W_{ij}[\text{argmax}(W_{ij})]$
iff $\text{argmax}(W_{ij}) \in W_{high}$ where $\forall p \in W_{high}$, $W[p] \geq \lambda$, $\lambda = \sum u_i v_i$.

The normalization factor divides the vector $W$ into two halves, the relations with weights greater than normalization factor will increase if $W_{avg}$ replaces $W_{ij}$. In other words, if the highest weight relation on the edge is among the higher weight relations

in the averaged constraint along the paths, then its weight is guaranteed to increase further. Contrary to this, when the highest weight relation on the edge is not the highest weight relation in the averaged constraint, a conflict takes place. Whether this conflict will lead to a decrease in the weight of the highest weight relation on the edge by replacing $W_{ij}$ by $W_{avg}$, is an obvious consequence of the above theorem. We formalize this condition as the following lemma.

**Lemma 2**: If $\text{argmax}(W_{ij}) \notin W_{high}$, then $W_{avg}[\text{argmax}(W_{ij})] < W_{ij}[\text{argmax}(W_{ij})]$.

Our premise is that as the weighted path consistency algorithm iterates, the weights in the network are adjusted based on the influences of the weights of the edges along the paths. Thus in an ideal situation (for eg a convex network), above lemma should not be satisfied at all. There are two possibilities here, the highest weight relation may exist in the averaged constraint with a lower weight or may be absent i.e. a weight of zero. In the first case, we solve the conflict by forcing the weight in the constraint resulting after intersection to be the highest value such that it becomes the highest weight relation for the next iteration. In the second case, as mentioned in the earlier section, the entropy of the constraint may or may not increase. In the latter case, a new atomic relation is forced to dominate other weights in the next iteration. For the cases when the next iteration highest weight value is less than the highest weight value in the current iteration, entropy will increase otherwise it will continue to decrease. We formalize these observations to introduce a concept of nasty constraints for IA networks.

**Definition 3**: A weighted IA constraint is said to be a *nasty constraint* if it satisfies either of the following conditions:

(a) If $\text{argmax}(W_{ij}) \notin W_{high}$ and atomic relation at $(\text{argmax}(W)) \notin R_{ij}$.

(b) If $\text{argmax}(W_{ij}) \notin W_{high}$ and atomic relation at $(\text{argmax}(W)) \in R_{ij}$.
and $W_{avg}[\text{argmax}(W)] < W_{ij}[\text{argmax}(W_{ij})]$

where $R_{ij}$ : constraint on edge *(i,j)* in the current iteration of weighted path consistency.

By the study of entropy of weighted IA constraints in the previous section, it is obvious that by the very definition of nasty constraint, entropy of a nasty constraint will increase when either of the above two conditions are satisfied. We formalize this consequence as following result.

**Theorem 5**: Entropy of a nasty constraint does not decrease monotonically over iterations of weighted path consistency.

A vector that is initially generated with all the components with equal values, this will correspond to maximum entropy of the vector. If the same vector is subjected to some operations in an iterative manner such that the value of one of the components goes on dominating all others, the entropy of this vector will go on decreasing assuming number of non-zero components do not change. A stage will come, beyond which entropy cannot decrease further and hence stabilizes. We exploit this observation in the next section as the termination condition of the algorithm proposed in this paper.

## 6 Approximate solution for IA networks

In this section, we propose a method to determine an approximate solution for IA networks based on our foregoing analyses. We propose an algorithm to identify nasty constraints in weighted IA networks and settle these to compute an early solution as shown in the pseudocode in Table 3.

***compute_approx_solution***(W(N))
Output: A singleton network $\tau$ that is a solution
*while* no solution *weighted_path_consistency_iteration*(W(N)) *enddo*
***weighted_path_consistency_iteration***(W(N))
$\forall W_{ij}$ $\forall k = 1$ to *n*, such that $k \neq i$ and $k \neq j$
$W(k) = W_{ik} \otimes W_{kj}$
$W \leftarrow$ normalized non-zero average over W(k)
$\lambda = \sum w[p]w_{ij}[p]$, p = 1 to 13
$W_{avg}[i,j] \leftarrow W \cap W_{ij}$
{where $\otimes$ and $\cap$ are weighted composition and intersection operators}
partition W such that $W=W_{high} \cup W_{low}$, $W_{high} \cap W_{low} = \varnothing$,
$\forall p \in W_{high}$, $W[p] \geq \lambda$, $\forall q \in W_{low}$, $W[q] < \lambda$
*if* $(\text{argmax}(W_{ij}) \notin W_{high})$
if $(IA(\text{argmax}(W_{ij})) \notin R_{ij})$ mark (i,j): nasty constraint endif
if $(IA(\text{argmax}(W_{ij})) \in R_{ij})$ and $(W_{avg}[\text{argmax}(W_{ij})] < W_{ij}[\text{argmax}(W_{ij})])$
mark (i,j) as a nasty constraint
endif
$W_{avg}[\text{argmax}(W)] = 1.0$
renormalize $W_{avg}$
endif
Replace $\forall(i,j)$ $W_{ij} \leftarrow W_{avg}(ij)$
$\forall(i,j)$ $\tau_{ij} \leftarrow$ atomic relation at argmax(W)
if $\tau$ is path consistent then solution found
$\forall(i,j)$ if $\tau_{ij} \notin C_{ij}$ then constraint is violated *endif*
where $C_{ij}$ is the disjunctive constraint in the IA network N
*endif*

**Table 3**. approximate solution algorithm.

Clearly *compute_approx_solution* is of $O(n^3T)$ complexity, if we assume that *T* number of iterations of weighted path consistency are executed to compute a solution. As per our foregoing analyses in the previous section, this algorithm captures those constraints as nasty constraints for which entropy fluctuates. It is observed that there are some more constraints that are not the nasty constraints, but still the highest weight relation along the paths is forced to become highest on the edge. These are those constraints for which a conflict takes place and the highest on the edge is not absent along the path, but has smaller weight. We term all the constraints (including nasty constraints) where any time this type of adjustment of weights takes place as *approximated constraints*(AC). The algorithm starts with the state of highest entropy for all the constraints, that corresponds to the starting point when all the atomic relations in a constraint are assigned equal weights. As the weighted path consistency algorithm iterates, *compute_approx_solution* ensures that entropy of every constraint

to decrease monotonically. In the later iterations, weight of an atomic relation dominates others, leading to the state of least entropy beyond which a bounded variable like entropy (with a minimum value of -1) cannot decrease. Over iterations of weighted path consistency, our algorithm reduces the number of inconsistent triplets in the singleton network by forcing the highest weight relation on the edge to agree with the one with maximum support along the paths. We claim that on termination, it will compute a solution. The solution may be an exact one for easy instances and an approximate one for hard instances. Thus our method is a complete method for determining approximate solution for IA networks.

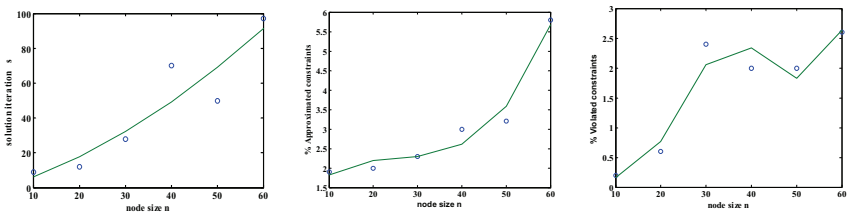
**Theorem 6**: *compute_approx_solution* is a complete algorithm.

## 7 Experimental analysis

The objective of the experimental analysis is essentially to confirm our theoretical analyses as discussed in the previous sections. Realizing the algorithm is complete, we attempt to determine the instances that are known to be consistent and completely cover both easy as well as hard problem regions of IA networks. The experiments are conducted on Windows based PC with 2GHz clock speed, 512 RAM and Visual C++ environment. We have experimented with 480 instances of known consistent general IA networks with $n$ in the range [10,60]. The graphs shown in Figure 4 indicate the performance of our method.

The model for instance generation is same as that proposed in [13]. The iteration at which a consistent scenario is obtained, ($s$) is noted and average of these is taken for each combination of $n$ and $d$. This approach gives an empirical estimate of average number of iterations required to get a solution for general IA networks. We make use of statistical regression models to analyze empirical results at arrive at the best-fitting curve. Figure 4(a) shows that the solution iteration depends on the constraint tightness. Figures 4(b) explains that higher the number of approximated constraints higher is the number of violated constraints. Our method is able to solve 100% of the problems for $n$ in the range [10,40]. Two instances for 40 nodes and five problems in 60 nodes set of problems are left without a solution, i.e. 95% of success rate. With experience, we say that this 5% of failure is due to numerical errors.

Any comparison of this method with backtrack algorithm will not be in place. We feel that comparison of two methods that give different types of solutions does not help us in this context. However, an outright advantage of our method can be simply seen by the fact that for 50 and 60 node problems, backtrack is known to take exponentially high computation time, where as our method gives the solution in a maximum computation time of 80 minutes, which is equivalent to a maximum of 1000 iterations of weighted path consistency algorithm. This method is able to solve even hard problems in reasonable time despite a large number of nasty constraints.




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

**Figures 4.** Linear cubic regression models.
**(a)** $s = 0.0136 + 0.7529n - 2.6n^2$
**(b)** $AC = 3.83 + 0.7475n + 0.0480n^2 - 0.0011n^3$
**(c)** $VC = 0.73 + 0.16n - 0.00594n^2 + 0.0001n^3$

## 8 Conclusions

The present work introduces a new paradigm for TCSP using entropy-based interpretation of IA as against the known method backtrack. We provide an insight into the well-known fact that convex networks are easy to solve. General IA problems with relations not belonging to any of the tractable classes are solved with help of a complete method. We provide here a linear time algorithm that captures the hardness of the problem in terms of nasty constraints, exploiting structure of individual problems. Our algorithm computes approximate solution for hard problems in polynomial time with exact solution a special case. In the process of handling the conflicts, the link with the original problem is not lost. It is possible for an interactive choice of nasty constraints to be settled, that may be crucial to the problem. It is possible to keep track of iteration-wise resolved atomic relations. User can analyze the impact of avoiding or choosing a new atomic relation. We propose to extend this study to propose PTAS with approximation bounds for general IA networks and overconstrained problems.

**References**

1. Allen: Maintaining knowledge about temporal intervals. *Communication of the ACM*, 26(11): 832-843, 1983.
2. Bhavani, S. D. and Pujari, A. K.: EVIA-Evidential Interval Algebra and Heuristic backtrack-free algorithm, *Constraints*, 9 (3), 2004, 193.
3. Boukeas, G., Halasis, G. Zissimopoulos, V. and Stamatopoulos, P.: Measures of Intrinsic Hardness for Constraint Satisfaction Problem Instances. LNCS 2932, 184-195, 2004.
4. Clark, D.A., Frank, J. Gent, I.P., Macintyre, E., Tomov, N. and Walsh, T.: Local Search and the Number of Solutions. *Proc. of CP-96*, Springer(1996), 119-133.
5. Crutchfield, J. Feldman, D.: Regularities Unseen, Randomness Observed: Levels of Entropy Convergence. Chaos 13(2003) 25-54.
6. Dechter: *Constraint Processing*. Morgan Kaufmann Publishers, San Francisco, USA, 2003.

7. Gent, I.P., MacIntyre, E., Prosser, P., Walsh, T.: The Constrainedness of Search. In: AAAI/IAAI. Volume 1. (1996) 246-252.
8. Gerevini, A. and Renz, J. Combining topological and qualitative size constraints for spatial reasoning. *Proc. of the 4th International Conference on Principles and Practice of Constraint Programming* 1998, Pisa, Italy.
9. Hogg, T., Huberman, B. Williams, C.: Phase Transitions and the Search Problem. Artificial Intelligence 81 (1996) 1-15.
10. Larrosa, J., and Schiex, T.: In the quest of the best form of local consistency for weighted CSP. In *Proc. of the 18th IJCAI* (Acapulco, Mexico, 2003), 239–244.
11. Ligozat, G.: A new proof of tractability for ORD-Horn relations. *Proceedings of AAAI-96*, 395-401.
12. Nebel: Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ORD-Horn class. *Constraints*, 1(3):175-190, 1997.
13. Pujari, A.K., and Adilakshmi, T.: A Novel Heuristic to Solve IA Network by Convex Approximation and Weights. In *Proc. of the 8th PRICAI* (Auckland, New Zealand, August 2004), 154-163.
14. R´enyi, A.: On measures of entropy and information. Selected Papers of Alfred R´enyi, 2(180):565–580, 1976a.
15. 18. R´enyi, A.: On the foundations of information theory. *ibid*, 3(242):304–318, 1976b.
16. 19. R´enyi, A.: Some fundamental questions of information theory. *ibid*, 2(174):526–552, 1976c.
17. Rossi, F., Venable K. B., Khatib, L., Morris, P. and Morris, R.: Two solvers for tractable temporal constraints with preferences. *Proc. AAAI 2002 Workshop on preferences in AI and CP Edmonton,* Canada.
18. Selman, B., Levesque, H. and Mitchell, D.: A new method for solving hard satisfiability problems. In Proceedings of the *Tenth National Conference on Artificial Intelligence*, 440-446, 1992.
19. Shannon, C. E. and Weaver, W.: The Mathematical Theory of Communication. Univ of Illinois Pr., 1963. ISBN: 0–252–72548–4.
20. Thornton, J., Beaumont, M., Sattar, A. and Maher, M. J.: Applying Local Search to Temporal Reasoning. *TIME,* 2002: 94-99.
21. van Beek and Manchak 1996: The design and experimental analysis of algorithms for temporal reasoning. *JAIR 4*:1-18.
22. Vilain, M. and Kautz, H.: Constraint propagation algorithms for temporal reasoning. In *Proc. Fifth National conference on Artificial Intelligence.* 377-382, Philadelphia, 1986.
23. Zitnick III, C. L.: Computing Conditional Probabilities in Large Domains by Maximizing R´enyi's Quadratic Entropy. PhD Thesis: CMU-RI-TR-03-20, 2003.

# Graphplan Based Conformant Planning with Limited Quantification

Alan Carlin[1], James G. Schmolze[1], Tamara Babaian[2]

[1] Department of Electrical Eng. And Computer Science
Tufts University, Medford, MA 02155 USA
{schmolze, acarli04}@cs.tufts.edu
[2] Department of Computer Information Systems
Bentley College, Waltham, MA 02452 USA
tbabaian@bentley.edu

**Abstract.** Conformant planners solve problems with a correct but incomplete description of the initial state, by finding plans that are valid for all possible assignments of the unknown atoms. Most conformant planners, however, do not handle universal quantification, which is a problem when the set of all domain objects is unknown or very large, and thus can not be enumerated. This paper introduces PSIGRAPH, a conformant planner that operates with universally quantified statements in the initial and goal states, as well as in the action preconditions. Thus, PSIGRAPH does not need to know the complete set of domain objects. We present the algorithm and the results of its experimental evaluation, showing that PSIGRAPH is competitive with other conformant planners. PSIGRAPH is based on Graphplan, but differs from previous approaches such as Conformant Graphplan in that it does not create multiple plan graphs.

## 1 Introduction

*Graphplan* [3] is a well-known and well-studied AI Planning algorithm. From a layer of initial conditions, it iteratively generates new layers of subsequent conditions that can result from actions, detects whether these subsequent conditions entails the goal, and if so, evaluates whether the path of actions that lead from the initial conditions to the goal is a valid solution. Graphplan repeats these three steps, extending the graph until a valid solution is found. A large amount of prior work has addressed the optimization of the closed-world Graphplan, as summarized in [14].

The solution extraction portion of Graphplan has proven to be the most computationally intensive, so optimizations have included memoizing unworkable solutions, as presented in the original Graphplan paper, forward checking to detect invalid solutions in advance, dynamic variable ordering [2], and formulating solution extraction as a *constraint satisfaction problem* (CSP). Variations on the latter approach [10] attempt to construct minimized explanations of why a solution is unworkable in the form of an unworkable set of propositions at a given time step, which we refer to as a *nogood*. These nogoods are stored, and future solutions are checked via an efficient algorithm [9] to detect whether they have a nogood as a


*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 65-76*

subset. If so, the solution is not explored further. Moreover, each nogood is *regressed* [10] to previous layers, to take further advantage of it.

The above optimizations have been implemented in closed-world Graphplan based planners. In closed-world planners, all propositions are assumed to be false unless otherwise noted. There are no unknown propositions. Recent work has explored the open-world problem, where some propositions are unknown. Conformant planners do no sensing and attempt to produce a single plan that will work in every contingency no matter what is unknown. Conformant Graphplan [13] is a Graphplan-based algorithm that produces a Graphplan for each possible world. More recent planners, including GPT [4], have expressed conformant planning as a search in a belief space. MBP [7] uses Binary Decision Diagrams [6] to represent belief states. CAltAlt-LUG [5] condenses multiple planning graphs into a Labeled Uncertainty Graph to conduct the search in belief space.

However, none of the above planners handle quantified information, or information about an infinite number of items. Finzi et al [8] produced an open-world planner, implemented as a theorem prover in the situation calculus, that could represent statements like "For all Blocks x, x is not on top of A", whereas the above planners would need to make a qualitatively different statement like (Clear A). We use an open world planning language called PSIPLAN [1] that can represent quantified statements about negated propositions. Furthermore, PSIPLAN can add exceptions to these statements, such as "For all Blocks x, x is not on top of A, except if x is Block B." Babaian and Schmolze called these statements *psiforms*. Exceptions to a psiform represent unknown information. That is, given the previous statement, the state of Block B being on top of A is unknown. PSIPOP [1] is a conformant partial order planner based on PSIPLAN.

This paper describes a new planner called PSIGRAPH, which implements a Graphplan based algorithm using the PSIPLAN language, and is thus able to act as a fast conformant planner for use in domains where quantification is needed.

In the next section, we review the original Graphplan algorithm, followed by a description of PSIPLAN. Afterwards, we explain PSIGRAPH, which combines the two. We then describe the methodology used in testing PSIGRAPH on the Blocks-World and Bomb-In-Toilet-with-Clogging (BTC) domains. Finally, we evaluate the results and draw conclusions.

## 2 The Closed-World Graphplan Algorithm

Graphplan constructs a layered, directed, acyclic graph. The first layer is assigned level 0, and the nodes in even numbered layers represent ground literals. The nodes in odd numbered layers represent operators. No literal or operator is represented more than once in a given layer. The initial conditions are assigned to nodes in layer 0. Letting the current layer of operators be called k where initially k=1, Graphplan repeats the following steps, increasing k by 2 each time, until it finds a solution.

- All possible operators, including maintenance operators (which simply *copy* a condition from one layer to the next condition layer) are assigned to layer k.

- For each operator in level k, Graphplan checks to see whether its preconditions are present on layer k-1. If not, the operator is removed from the graph. If so, a directed edge is created from each precondition on level k to the operator.
- The effects of the operator are added to layer k+1, with directed edges from the operator to these effect nodes.
- When all operators have been examined, mutexes are created between pairs of operators that cannot co-occur. For example, a mutex would occur between two operators whose preconditions are mutually exclusive.
- Next, mutexes are created between inconsistent pairs of conditions on level k+1.
- After all mutexes have been added, Graphplan evaluates whether layer k+1 entails the goal. If so, it is possible that Graphplan has found a solution. In the next phase, called Solution Extraction, Graphplan starts with the goal conditions from layer k+1 and checks to see whether there exists a set of edges from non-mutex actions that produce them. If so, Graphplan recursively checks to see whether these actions have non-mutex conditions which produce them. If the recursion reaches the initial layer, which by definition has no mutexes, then Graphplan has found a solution.

## 3 PSIPLAN

PSIPLAN [1] is an expressive language designed for open world domains. It offers limited quantification and tractable, complete reasoning. A database in PSIPLAN consists of ground literals and psiforms, the latter of which express possibly quantified *negative* information. Such quantification makes a database much more compact since there are often many more false facts than true ones. For example, a briefcase may have a pencil in it but there may be many things not in the briefcase. Moreover, for infinite domains, or for finite domains where some objects are unknown to the planner, quantification is essential. Consider the impossibility of stating that there is nothing in the briefcase except a pencil if the domain is infinite, or if the domain is finite but the planner cannot name all the objects in it. In both cases, one cannot enumerate all ground instances of ~In(x,B) .

To state that briefcase B has nothing in it except possibly pencil P in it PSIPLAN uses a *psiform* [~In(x,B) except x=P]. .Here the x is a universally quantified variable and ~In(x,B) represents that no x is "in" B. The *exception* x=P means that ~In(x,B) is not necessarily true when x=P. [~In(x,B) except x=P] is equivalent to the standard first order sentence $\forall x.\neg In(x,B) \vee x = P$. Combined with the atom In(P,B), it implies that P and nothing else is in B.

Psiforms are even more general in two ways. First the main part, which is the part before the word "except", can be a clause of negated literals. For example, [~In(x,B) or ~Pencil(x)] states that "for all x, x is either not in B or x is not a pencil" -- i.e., there are no pencils in B (though there might be other things in B). Second, the exceptions can themselves be "quantified" in that a set of ground clauses can be excepted. For example, [~InDir(x,y) or ~TexFile(x)] states that "for all x and y, either x is not in directory y or x is not a Tex file," which is equivalent to saying that Tex files are not in any directory. But this is odd. A more reasonable statement might be [~InDir(x,y) or ~TexFile(x) except y=/tex], which states that Tex files are not in any directory except possibly the directory /tex.

In some domains, one can use tricks to represent quantified information without explicit quantification, such as the use of Clear(x) in the blocks world. But the use of Clear depends crucially on the requirement that there is at most one block on top of another. In the briefcase example, we cannot use a trick such as Empty(x) because a briefcase can have 0, 1, 2 or more objects in it. We would need Empty(x), Empty1(x) to represent that x is empty except for 1 object, Empty2(x), etc.

The reasoning algorithms for psiforms include entailment, logical difference and logical image. Entailment is needed because we now have quantification. For example, if our goal is that from above, namely that no block be on B, [~On(x,B) or ~Block(x)], we can satisfy this with nothing being on B, [~On(x,B)], or with nothing being a block, [~Block(x)]. Logical difference lets us "subtract" one psiform from another to see what is not entailed. For example, P1=[~On(x,B) or ~Block(x)] "minus" P2=[~On(x,B) except x=A], which states that nothing is on B except possibly A ,, yields P3=[~On(A,B) or ~Block(A)], i.e., to entail P1 using P2 we must also have P3. Image is the complement of difference. The image of P2 on P1 is the subset of P1 that is entailed by P2, which is P4=[~On(x,B) or ~Block(x) except x=A]. All three types of reasoning are used in planning.

Formally, a PSIPLAN database is a set of ground literals and/or psiforms. A psiform is $P=[\sim P_1(x)$ or … or $\sim P_m(x)$ except $\sigma_1, \ldots, \sigma_n]$ where $x$ is possibly a vector of variables, $M(P)=[\sim P_1(x)$ or … or $\sim P_m(x)]$ is called the *main form* and $E(P)=\{\sigma_1,\ldots,\sigma_n\}$ are the exceptions. Each $\sigma_i$ is a substitution that binds a (not necessarily proper) subset of the vector of variables, $x$, to constants. The meaning of a psiform P is the conjunction of the clauses in $\phi(P)$. When P has no exceptions, $\phi(P)$ is the set of all ground instantiations of P, i.e., $\phi(P)=\{M(P)\sigma \mid M(P)\sigma$ is a ground clause$\}$. Otherwise, $\phi(P)=\left(\phi(M(P)) \setminus \left(\phi(M(P)\sigma_1) \cup \ldots \cup \phi(M(P)\sigma_n)\right)\right)$ where \ is set difference.

A ground clause C1 entails another ground clause C2, written C1|=C2, if and only if the literals in C1 are a subset of the literals in C2. A psiform P1 entails a psiform P2, P1|=P2, if and only if every clause in $\phi$(P2) is entailed by some clause in $\phi$(P1). The *image* of P1 onto P2, written $P1 \rhd P2$, is the subset of $\phi$(P2) that is entailed by P1. Thus $\phi(P1 \rhd P2)=\{p \mid p \in \phi(P2)$ and $\phi(P1)|=p\}$. Finally, the *e-difference* (i.e, logical difference) of P2 minus P1, written P2-P1, is the subset of $\phi$(P2) that is not entailed by P1. Thus $\phi$(P2-P1)$=\{p \mid p \in \phi(P2)$ and $\sim(\phi(P1)|=p)\}$. $(P1 \rhd P2)$ and (P2-P1) partitions $\phi$(P2). We note that image and e-difference can be represented by a set of psiforms, and that all three operations -- entailment, image and e-difference -- require time and space that is polynomial in the size of the database under certain reasonable assumptions [1] which we make in this paper.

# 4 PSIGRAPH

We split the description of PSIGRAPH into four parts: the definition of a planning problem, the overall algorithm, graph generation, and solution extraction.

## 4.1 Definition of a Planning Problem

PSIGRAPH is given the following:

- A set of initial conditions, which consists of ground literals and/or psiforms

- A set of goals, which consists of ground literals and/or psiforms.
- A set of operators, each of which consists of:
  - a name, which specifies the variables in the operator structure.
  - a set of preconditions, which consists of literals and/or psiforms.
  - a set of effects, which consists of literals.

In the currently implemented version of PSIGRAPH, we do not allow conditional effects and disjunctions are limited to psiforms.

The overall PSIGRAPH algorithm is in Figure 1 and is the same as the closed-world Graphplan algorithm.

### 4.2 Graph Generation

The graph generation portion of PSIGRAPH is based on that of the closed-world Graphplan in that each precondition of each operator is checked to see if it is entailed in the previous layer. If all of the preconditions for the operator are so entailed, the operator is retained and the effects of the operator are generated for the next layer. Otherwise the operator is removed from the graph. However, there are three issues presented by the use of psiforms in the PSIGRAPH domain.

(1) Preconditions may be *nearly entailed* by propositions.
(2) There may be more than one way to entail a precondition.
(3) Generated psiforms on the next layer may only be *partially mutex* with other generated psiforms, and this will make future reasoning difficult.

We explain each of these in turn. But first, we say that a psiform P1 *nearly entails* another psiform P2 if and only if the main part of P1 entails the main part of P2, ignoring exceptions, i.e., P1 nearly entails P2 iff M(P1)|=M(P2).

```
Algorithm PSIGRAPH
    Current-Level = Initial-Conditions; Iterations=0
    Repeat
        Iterations++;
        Next-Level = Generate-New-Layer(Current-Level);
        If Find-Plan(Next-Level) == SUCCESS
            then Return(SUCCESS);
        End if
        Next-Level = Current-Level;
        If iterations > MAX_ITERATIONS, Return(FAIL);
    end Repeat
```

**Fig. 1.** The overall PSIGRAPH algorithm.

### 4.3 Multilinks

The first issue arises when a combination of two or more propositions from a layer entail a precondition or goal, but neither by itself is sufficient for such entailment. For example, let a precondition state that block B is clear of anything on top, i.e., [~On(x,B)], and let the previous layer include the propositions:

[~On(x,B) except x=C, x=D], ~On(C,B), ~On(D,B)

Together, these three conditions entail the precondition. In such a case, PSIGRAPH draws a *multilink* between the operator and the three prop ositions. A multilink in PSIGRAPH acts just like a link or an edge in Graphplan. It is a set of edges from one or more propositions on layer k to an operator on layer k+1. As a plan proceeds these edges must be followed atomically, that is, all at once or not at all. Note that the closed-world Graphplan may be viewed as a form of PSIGRAPH where all the multilinks have exactly one edge.

### 4.4 Finding the complete set of Multilinks

The second issue is that a precondition may be entailed by more than one multilink. For PSIGRAPH to be complete, it must find all possible multilinks. Thus it implements the function Satisfy-Goal, which returns the set of sets of propositions in a given layer where each set, taken together, entails a given goal. It *e-subtracts* each potentially helpful proposition from the goal, and recursively calls itself to satisfy the remainder. The algorithm is in Figure 2 where \ is set subtraction and – is e-difference.

The first argument to the recursive call is the union of the set of goals without G and the e-difference of G minus P. The latter is the portion of G that is not entailed by P. In general, e-difference returns a set of psiforms.

```
Function Satisfy-Goal (Goals, Props, Sofar)
- Goals is a set of psiforms to achieve.
- Props is the set of conditions to examine.
- Sofar is the current partial solution set.
If Goals is empty then return {Sofar}
        // Return a set whose only element is the set Sofar.
else Let Result = {}
                For each P in Props
                        For each G in Goals
                                If P nearly entails G
                                then Result=Result U
                                        Satisfy-Goal((Goals\G) U (G-P),
                                                Props\P, SoFar U {P})
                        end inner for
                end outer for
        return Result
end If
end Function
```

**Fig. 2.** Satisfy-Goal

The first call for a given Goal is: Satisfy-Goal({Goal}, Props(Layer), {})
where {Goal} is a singleton set containing Goal, Props(Layer) is the set of propositions in the Layer and {} is the empty set.

### 4.5 Partially Mutex

Just like Graphplan, PSIGRAPH generates all of an operator's effects on the next layer. For negated literals, determining mutexes between conditions on this next layer is the same as Graphplan since all literals are ground: If A is an atom, mark as mutex the pairs A and ~A. With psiforms, mutexes are more complicated because an atom A might be inconsistent with only *part* of a psiform P, i.e., P might entail many ground clauses where A is inconsistent with only some of them. For example, A=On(A,B) is inconsistent with P=[~On(x,B)] but P entails many ground literals besides the one that is inconsistent with A. We cannot mark A and P as mutex because it is an overgeneralization and may prevent finding some solutions. Instead we *split* P into two parts: P1, which represents the subset of P that directly conflicts with A, and P2, which is the remainder of P. In the above example, we split P into P1=[~On(A,B)] and P2=[~On(x,B) except x=A].

The above is accomplished using the image and e-difference operation described earlier. An atom A is inconsistent with a psiform P iff P⊨(~A). If not, there is no mutex. If so, we calculate P1=([~A] ⊳ P) and P2=(P1-[~A]). Remember that P1 and P2 are sets of psiforms, and we note that P1 must be a singleton set. If P2 is empty then no splitting occurs because [~A] entails all of P. In this case, A and P are simply marked mutex. If P2 is not empty, then node P is replaced by P'=(P1 U P2) in the graph and A is marked mutex with the single psiforms in P1. P1 and P2 inherit the uplinks from P. Their downlinks are easily recalculated from P's downlinks.

### 4.6 Solution Extraction

Solution extraction of PSIGRAPH follows the algorithm of Kambhampati [10] by using Explanation-Based Learning (EBL) and Directed-Backtracking (DDB). Several issues that arise due to the use of psiforms in PSIGRAPH require only minor modifications to the algorithm

(1) There may be more than one set of propositions that entails a goal or precondition.
(2) A set of propositions may be mutex, even though there is no pairwise mutex. We refer to these sets as nogoods.

The first issue is solved merely by following all possible multilinks backwards during backtracking. Although this increases the search space, the EBL/DDB algorithm is extended to mark additional sets of unreachable propositions as memoizations of nogoods. The only difference is that in PSIGRAPH, a failed solution could return more than one conflict set. Each conflict set is stored as a memo and regressed. The memo sets are stored in a UB-Tree [9].

The second item above refers to disjunctive psiforms. A disjunctive psiform may be mutex with a pair of atoms taken together, while being mutex with neither separately. These sets are detected at graph generation time by scanning the layer for sets of atoms each of which is mutex to a term in the disjunction. They are stored as nogoods in the UB-tree.

## 5 Evaluation

PSIGRAPH was implemented in Allegro Common Lisp, and tested on the BTC (bomb in toilet with clogging [11]) and Blocks-World domains. For the Blocks-World domain, we generated problems using the BWStates program [12] and recoded them in PSIGRAPH. For BTC, we rephrased the initial conditions as follows. In this example, there is one toilet, T1, and 2 packages, P1 and P2.

[~Package(x) except x=P1, x=P2],
Package(P1), Package(P2), Toilet(T1), ~Clogged(T1)

The first proposition states that nothing is a package except possibly P1 and P2. We also rephrased the goal.

[~Package(x) or ~Armed(x)]

i.e., every x is either not a package or not armed. The Dunk(P,T) action had preconditions Package(P) and ~Clogged(T), and effects Clogged(T) and ~Armed(P). The Flush(T) action had no preconditions, and effect ~Clogged(T).

We also performed experiments where it was not known whether the toilet(s) were clogged (i.e, we removed ~Clogged(T1), etc., from the initial state), and the effect was small. We will soon see that PSIGRAPH is not sensitive to this type of change in the initial state. We ran our experiments on a 2.4Ghz Dell Linux workstation.

We used two different versions of PSIGRAPH. The first, PG1, performed an exhaustive solution extraction search on each layer before failing and proceeding to the next layer. PG1 always finds an optimum parallel solution. The second, PG2, differs from PG1 in the following ways:

- (Mod 1) All pairs of non-maintenance actions were labeled mutex.
- (Mod 2) Solution extraction failed after n nogoods were found, where n is the number of operators in the domain, unless the number of planning layers was at a theoretical maximum (in which case solution extraction failed). The last solution extraction performed before PSIGRAPH gives up is always a complete search.
- (Mod 3) Solution extraction was only attempted every fifth layer.

(Mod 1) means that PG2 finds only linear plans. Problem BTC(40,6) requires 13 time-steps under PG1 and 81 timesteps under PG2, although both have the same number of non-maintenance operators.

(Mod 2) prevents the planner from getting bogged down in solution extractions that are likely to fail. As a result, it may return non-optimal plans. But as long as the last attempt is a full solution extraction, it will never fail to solve a plan because of (Mod 2). This is because solution extraction works just fine on overly long graphs. BTC was not assigned a theoretical maximum, but the Blocks-World domain has a maximum number of plan steps of 2 times the number of blocks

(Mod 3) has the same intention as (Mod 2).

Table 1 shows our results in BTC 1-toilet problems. BTC 1-toilet results have been published for other conformant planners, and a summary in [5] includes results for CAltAlt-Lug [5], HSCP, GPT [4], and CGP [13]. The summary shows HSCP as the fastest timing on this domain, taking 98 seconds for the 20 package problem, 674 seconds for 40 packages, and 5100 seconds for 60 packages. We note that these planners allow conditional effects but not quantified information, whereas PSIGRAPH does not allow conditional effects, but does allow limited quantified information. The effect is that the difference in expressiveness helps make BTC an easier problem for PSIGRAPH, as the DUNK action has no preconditions that need to be explored.

Table 2 compares the BTC 10-package 3-toilet problem (BTC(10,3)) and BTC(40,6), where the possible clogging of all toilets was unknown, to published results of WSPDF [8], who used a 333 MHz Sun Sparc 10 Ultra workstation. Finzi et al use a domain dependant BadSituations marker to limit their search space. In WSPDF, a BadSituation occurs when a toilet is flushed twice without an intervening dunk, when a package is dunked when there is an undunked package lower in number, and when a toilet is flushed when there is an unflushed toilet lower in number. We did not use the

**Table 1.** Timings of various planners on various domains. Times are in seconds. PSIGRAPH was run 5 times on a 2.4 Ghz Pentium processor. All other results come from [5] on a 2.66 Ghz Pentium 4. Times are in format (x/y), x is in seconds, y is in plan steps. All plans in the same row produce the same number of plan steps, unless otherwise noted. * indicates no solution

| Domain | PG1 | PG2 | Caltalt Lug | HSBP | CGP |
|---|---|---|---|---|---|
| BTC (20,1) | 2.46/39 | 1.7/39 | 651 | 98 | 465/3 |
| (40,1) | * | 14.4/79 | 8009 | 674 | * |
| (60,1) | * | 80.2/119 | 38393 | 5100 | * |

**Table 2.** Timings in the BTC domain for multiple toilets with high uncertainty. PG1 and PG2 were run using a 2.4 Ghz Pentium 4 processor. WSPDF is reported from [8] on a 333 Mhz UltraSparc 10. * indicates no solution.

| Domain | PG1 | PG2 | WSPDF | |
|---|---|---|---|---|
| BTC(10,3) | 12.5/7 | 1.2/19 | .32/20 | |
| BTC(40,6) | * | 80.3/79 | 114/80 | |

**Table 3** . Averaged results of running PSIGRAPH on 10 random examples in the BW domain. Domains are of the form BW(a,b) where a is the number of blocks and b is the number of blocks whose location is unknown. Results are of the form x(y)/z, w here x is mean time in seconds, y is mean plan steps,. And z is the maximum number of propositions found in a single layer. '*' indicates a trial had no solution found after 10 minutes, '-' indicates the experiment was not run

| Domain | PG1 | PG2 |
|---|---|---|
| BW(8,0) | 3.9(3.6) / 136 | 21.6(4.3) |
| BW(10,0) | 28.2(5) / 210 | * |
| BW(11,0) | 81.7(5.7) /253 | * |
| BW(12,0) | 334.1(4.1)/300 | * |
| BW(15,5) | 37.8(4.4) /210 | - |
| BW(20,10) | 54.1(5.2)/210 | - |

.

above domain restrictions but we did try to limit the search space in PG2 (see above).

As the results show, PG1 produces optimal solutions, even on multiple-toilet problems. This is because it performs a complete search of the solution space. Its

disadvantage is that it spends large amounts of time performing failed solution extractions, and this is enough to make the planner time out for large problems.

PG2, by contrast, finds solutions much faster. The speed of PG2 is in part an artifact of the simplicity of the BTC domain, as PG2 does not need to spend much time at all in solution extraction. In these experiments, PG2 was dominated by the graph generation phase, a trend that would reverse itself on more difficult problems. Graph generation is a comparatively easy task whereas solution extraction requires searching an exponential number of possible solutions. Furthermore, if the metric of finding n mutexes per attempt at solution extraction (Mod 2, where n is the number of literals on the layer) makes little progress on each iteration, PG2 might take longer than PG1. Also, PG2 relies on the hope that it will find a solution without exploring the whole search space. We ran PG2, for instance, reversing the order that the operators are considered (that is, we tried preferring maintenance actions instead of preferring non-maintenance actions), and PG2 showed the same difficulties for larger BTC domains as PG1. Thus, PG2 may prove to be fragile on other domains. The results above for PG2 should be viewed as an optimistic scenario, not the expected scenario. PSIGRAPH is presented with a similar dilemma to that faced by a closed-world Graphplan with a large number of propositions. We note that the BTC domain will generate approximately 2*P propositions per layer, where P is the number of packages, as there are P initial conditions of the form (Package P), and P exceptions to [~Package(x)].

We ran PG1 and PG2 on the Blocks-World domain to test the algorithm in a more difficult domain as well as to test its sensitivity to the number of unknowns in the initial state. We used the BWStates program [12] with various numbers of blocks with various numbers of unknown locations. Each problem was translated to PSIPLAN, including elimination of Clear and use of psiforms instead. Table 3 shows the tradeoff between PG1 and PG2.

Table 3 shows that PG1 is better on domains like blocks-world, presumably because in the blocks-world doing an exhaustive solution extraction early and often is a good idea since more mutexes will be found anyway. It also shows that PSIGRAPH is relatively insensitive to unknowns in the domain. Domains (15,5) and (20,10) are comparable to (10,0), in that both have the same number of known facts in the initial state. Unknowns will not affect solution extraction; they only affect the time taken for graph generation as they increase the number of operators to check. It should be noted that Finzi et al. also timed their WSPDF theorem prover on the Blocks-World domain, with a domain-specific BadSituation() predicate which favored exploration of good towers. Their planner produced 17-step plans in 31.2 seconds, with an additional 50.1 seconds to compile the domain for 20 blocks and 10 unknowns on a 333 Mhz UltraSparc processor. These results are roughly comparable to ours. However, PSIGRAPH did not rely on any additional domain information, like the BadSituations..

## 8 Conclusion

We introduced PSIGRAPH, a conformant planner based on Graphplan that uses the PSIPLAN language, which allows for limited quantification. PSIGRAPH can work in

infinite domains, and in finite domains where not all objects are known and admits very compact representations of domains with a large quantity of negative facts.

We evaluated PSIGRAPH on the BTC and Blocks-World (BW) domains, and compared results from other planners. Only one of these other planners allows quantification, namely WSPDF of [8]. For several BTC problems, PSIGRAPH is faster than most other planners tested. In BW, PSIGRAPH is comparable to WSPDF, though it is not clear how WSPDF's domain dependent BadSituations affected its timings.

Future work will improve PSIGRAPH, investigate more domains, and develop a better understanding of the differences between PG1 and PG2. We will also expand PSIGRAPH to allow conditional effects and general disjunction in the initial state, and will explore the use of binary decision diagrams [6] for both ground and quantified formulas.

## References


1. Babaian, T. and J. Schmolze (2000). PSIPLAN: open world planning with psi-forms. In Artificial Intelligence Planning and Scheduling: Proceedings of the Fifth International Conference (AIPS'00), pages 292-300.
2. F. Bacchus and P. van Run (1995). Dynamic variable ordering in csps. In Proceedings of the 1995 conference on Principles and Practice of Constraint Programming pages 258-275, September 1995.
3. Blum and M. Furst (1995). Fast Planning through planning graph analysis. In Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95), pages 1636-1642, 1995.
4. Bonet and H. Geffner. Planning with Incomplete Information as Heuristic Search in Belief Space. In AIPS-2000, pages 52-61, 2000.
5. Bryce, D., Kambhampati, S (2004)., and Smith, D.E. AAAI 2004 workshop on Learning and Planning in Markov Decision Processes, 2004.
6. Bryant, R.E. (1986). Graph-Based Algorithms for Boolean Function Manipulation. IEEE Transactions on Computers, C-35(8), 677-691.
7. Cimatti, A. and Roveri, M. (1999). Conformant planning via model checking. In Biundo, S. ed., Proc. ECP99: European Conference on Planning. Springer-Verlag.
8. Finzi, Pirri, and Reiter (2000). Open World Planning in the Situation Calculus. In Technical Report, University of Toronto. AAAI, pages 754-760, 2000.
9. Hoffman and Koehler (1999). A new Method to Index and Query Sets. In 16th IJCAI, pages 462-467, 1999.
10. Kambhampati (2000). Planning Graph as a (Dynamic) CSP: Exploiting EBL, DDB and other CSP Search Techniques in Graphplan. Journal of Artificial Intelligence Research 12 (2000), pages 1-34.
11. McDermott, D. A critique of pure reason. Computational Intelligence, 3(3):151-237, 1987.
12. Slaney, J., and Thiebaux, S. 1996. Linear time near-optimal planning in the blocks-world. In Proc. Thirteenth National Conf. on Artificial Intelligence, 1208-1214.
13. David E. Smith and Daniel S. Weld. Conformant Graphplan. In (AAAI-98) and IAAI-98), pages 889-896, Menlo Park, July 26-30, 1998. AAAI Press.
14. Weld, D. S. (1999). Recent advances in AI planning. AI Magazine 20(2), 93-123.

# Multiagent Systems and Distributed AI

# A Distributed Multiagent Workflow System

César A. Marín and Ramón F. Brena

Centro de Sistemas Inteligentes
Tecnológico de Monterrey, Campus Monterrey
Eugenio Garza Sada 2501
C.P. 64849. Monterrey, Mexico
{cesarmp, ramon.brena}@itesm.mx

**Abstract.** The decentralized and distributed nature of workflow in organizations demands for support from decentralized and distributed computational systems. However, most conventional workflow applications use centralized architectures. Agent technology seems to be an adequate approach for supporting distributed systems. We have extended the capacities of a multiagent system for knowledge and information distribution in such a way that it can handle general workflow processes in a decentralized way. A working prototype is reported, and quantitative experiments have been conducted to show that the distributed workflow process flow control makes possible better scalability than the centralized counterpart.

## 1 Introduction

Within enterprises, streamlining processes have led to the implementation of paperless document circulation by means of *workflow* management systems (WfMS) [1, 2]. They are today a standard component of many enterprise-wide information systems and their value is widely acknowledged.

Within commercial and industrial domains, the business process execution and the process flow control are performed in a decentralized way because organizations are physically and often logically distributed. In other words, there is no central entity orchestrating each activity composing the whole business process. This decentralized and distributed nature of workflow in organizations demands for support from decentralized and distributed computational systems. However, most conventional workflow applications use centralized architectures.

In this paper we present an extension of an Information and Knowledge distribution system [3–5], which is an agent-based information system aimed to distribute the right piece of knowledge to the right person within different parts of an organization. In fact, distributing knowledge and information items could be thought of as a restricted kind of workflow, as it just comprises a document generation and its distribution, ending with the document reception by a final user. But if, for instance a document needs to pass through authorization in order to be distributed, then a more complex workflow is needed, and even this simple task is beyond the basic version of our knowledge distribution system. So, we enhanced our knowledge distribution system with general workflow capabilities.

© *A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 79-88*

This paper structure is as follows: After this introduction, we present some background about our knowledge distribution multiagent system. Then, in section 3 we present our proposal. In section 4 a working prototype is presented, which is validated experimentally in section 5. Then, we compare our work with others in section 6, followed by a conclusion.

## 2 Background - Our System Architecture

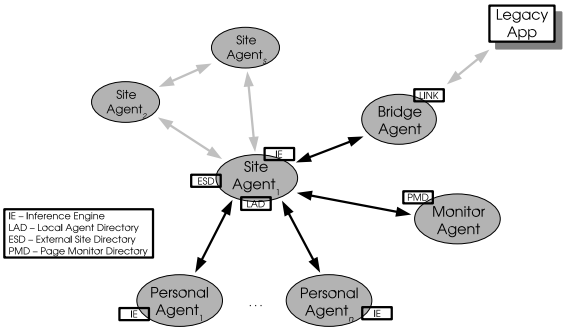



**Fig. 1.** Knowledge distribution system architecture.

Our workflow system is an extension of an information distribution system [6], which is based on a multiagent architecture shown in Fig. 1. It comprises some types of agents which appear in the mentioned figure but from which we are going to explain just those agents which are important for the work we are presenting:

**Site Agent**. This agent, works like a network router; it receives messages from any agent and distributes the information to the proper users under its site or domain. The distribution is made by first finding the corresponding users located in conceptual hierarchies. These hierarchies may represent organizational departments, interest areas, work groups, etc. Each Site Agent keeps in touch with others Site Agents so that they all together make a network of agents for information distribution.

**Personal Agent**. Each user may have one personal agent that filters the information addressed to the user and shows it through a web browser, sends him/her an e-mail or a message by a SMS service.

## 3 The Proposed Architecture

In our architecture, we are going to take the "personal agent" of JITIK as the basic workflow executers. The proposed solution for decentralized workflow process management consists in breaking down the workflow process execution

and the process flow control into small execution units handled by intelligent agents, and allowing the agents to reflect the organizational structure and the way processes are controlled and executed, i.e., distributed and decentralized.

For this purpose, two agent types are required: a new agent type named *Registry Agent* for holding process descriptions, keeping track of all the running processes at every moment and creating process instances on demand; and the existing *Personal Agent* for assisting its user/worker to perform his/her assigned tasks. In the end, Personal Agents are the actual organizational processes orchestrators in a distribution and decentralized fashion.

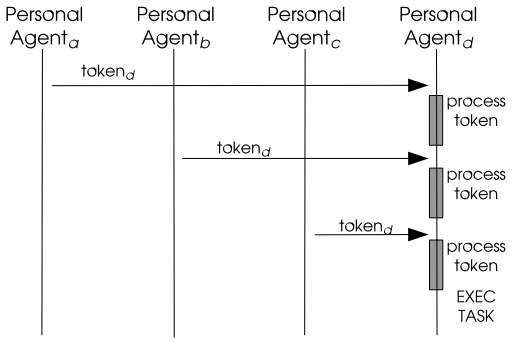
Once all agents (one Registry Agent and one Personal Agent for each user participating in a process) are up and running, the Registry Agent receives a process description [7] as input and segments it into atomic task descriptions. A *task description* is composed by the process identifier this task belongs to; the information to be handled which can be a link to a document; the assigned Personal Agent referenced by its user description in terms of the organization, i.e., the Personal Agent of the user in department $D$ and position $P$; a list of tasks to be enabled right after this task finishes its execution containing the corresponding Personal Agent reference; the join and split operations to apply; and the number of flows that converge to this task. After process description segmentation, the Registry Agent distributes each task description to the corresponding Personal Agent executor. This way, all Personal Agents know what to do in advance when a task of a process instance is running, resembling the way an organization works. It is assumed that each task is assigned to only one user, i.e., a unique Personal Agent.

### 3.1 Agent Communication

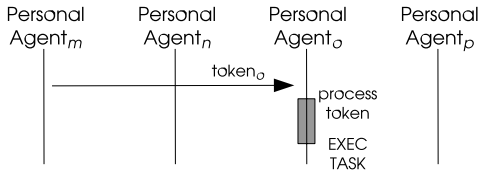
Since all tasks are distributed, Personal Agents need to send messages among them in order to enable tasks of the same process instance. In Petri Nets, a *token* is a marker that specifies in which part of the net is occurring the actual processing. In our system, a token is an agent message which contains a process ID, an instance ID and a task ID over which the message recipient must operate.

A task is enabled when its Personal Agent receives the necessary tokens for task enabling according to a join operation (AND, OR, XOR), e.g. let us assume that in a process, tasks $t_a, t_b, t_c$ and $t_d$ exist and are owned by Personal Agents $PA_a, PA_b, PA_c$ and $PA_d$ respectively, and $t_a, t_b$ and $t_c$ are direct predecessors of $t_d$ which in turn synchronizes the three incoming flows, i.e., $PA_d$ must perform an AND-join operation in order to enable $t_d$. Therefore, right after $PA_a, PA_b$ and $PA_c$ finish its task execution, each of them send a token to $PA_d$. And only when all three incoming tokens are received task $t_d$ is enabled and ready for execution. A sequence diagram showing this token passing is illustrated in Fig. 2(a).

When a Personal Agent finishes a task execution and is about to enable the successive tasks in the process flow, it sends a single enabling token for each successor task to its owner Personal Agent according to a split operation (AND, OR, XOR), e.g., let us assume that in a process, tasks $t_m, t_n, t_o$ and $t_p$ exist




(a) AND-join operation.




(b) XOR-split operation.

**Fig. 2.** Enabling workflow tasks.

and are owned by Personal Agents $PA_m, PA_n, PA_o$ and $PA_p$ respectively, and $t_n, t_o$ and $t_p$ are direct successors of $t_m$ which in turn selects one of the three outcoming flows, i.e., $PA_m$ must perform an XOR-split operation in order to enable only one of $t_n, t_o$ or $t_p$. A sequence diagram showing this token passing is illustrated in Fig. 2(a), here, task $t_o$ was selected and thus the token was sent to $PA_o$.

A token can be sent by the Registry Agent or a Personal Agent. When the Registry Agent enables one or more tasks is because a process instance has just been created by it and the first tasks in such instance process are being enabled. This is the only case in which the Registry Agent is involved in the process flow control. When a Personal Agent enables one or more tasks is because it just finished the execution of one of its tasks. Notice that several tokens can be sent at a time by each Personal Agent for different process instances. Moreover, when a task status changes (e.g. from enabled to in-execution), the task owner sends a message to the Registry Agent to inform the event. This is for monitoring purpose and will not be explained here.

## 4 Prototype

The developed prototype for distributed and decentralized workflow process execution consists in several software layers shown in Fig. 3.

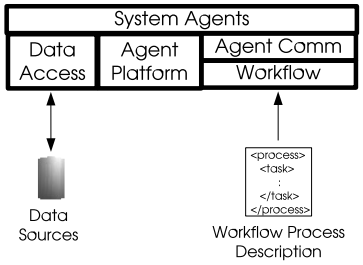



**Fig. 3.** Agent-based workflow software layers.

**Agent Platform**. The chosen agent platform for developing and executing our system agents was JADE because of its robustness [8]. Additionally, we used the JADE ACL messaging mechanism for agent communication.

**Data Access**. This layer is used for information access support. It allows agents to acquire information about their user or search other users' Personal Agents.

**Workflow**. Workflow process descriptions [7] taken as system input, are parsed and then segmented into atomic task description. This way, Personal Agents are able to know their assigned activities in advance and perform their tasks when a process instance is generated. This layer also works as information provider to the upper layer as explained below.

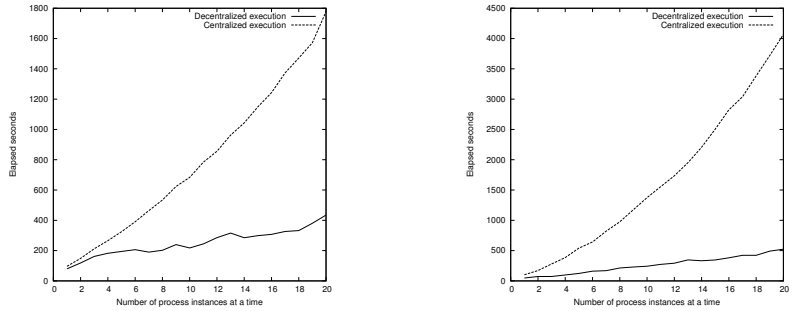
**Agent Communication**. On top of the Workflow layer the communication components were developed. These components are used for translating task descriptions into Tasks, as objects, so that Personal Agent can manage them. Based on these Tasks, Tokens can be generated and passed among agents for workflow enactment.

**System Agents**. Personal Agents, a single Registry Agent (and other system agent) are running constantly in the platform; they acquire information about users, such as who and where is his/her Personal Agent, through the Data Access layer; they rely on the Agent Communication layer for process instance creation, token passing and task enabling; and furthermore, the Registry Agent creates process instances and keeps track of all active processes.

## 5 Experiments

Since it is well known that a distributed application (e.g. using agents) diminishes the workload among its element while increases the communication, the objective of the experiments is to demonstrate that the proposed decentralized execution of workflow processes can be implemented (concept proof) and that performs better than a centralized approach. Thus, the performance of both approaches were compared using the elapsed time for executing certain number of process instances at a time.

Three processes were designed for this purpose, each of them representing some basic workflow patterns [9]. It was decided to test using the basic workflow




(a) Tasks in sequence (b) AND split and join

**Fig. 4.** Comparison between the centralized and the decentralized versions of workflow

patterns since when combined they form complex workflow processes. The first testing process represents the sequence pattern which, according to its nature, was combined with no other workflow pattern. And the second testing process represents the combination of the parallel split and the synchronization workflow patterns since they match, i.e., they are the AND split and joint.

An experiment consisted in the creation of an increasing number of process instances at a time, i.e., first one process instance, then 2, then 3 and so on up to 20 instances at a time. For each block of instances, the seconds elapsed from the first instance creation until the last task in terminate of the last process instance was measured. Thus, at certain moment there were several process instances running at a time.

The execution of a task consisted on waiting certain amount of simulation cycles. All tasks were standarized to 5 cycles and each cycle lasts 10 milliseconds, which means that the total amount of time for executing a task is 50 milliseconds. Notice that there was no central control on the passing time, i.e., each agent had to decide how much time had elapsed by its own.

For resembling the centralized approach, all tasks of the testing processes were assigned to one single agent who had to perform the whole work by itself. There were other threads along with the centralized version in order to allow all agents to operate under the same conditions.

For simulating the distributed approach, all tasks of the testing processes were completely distributed, i.e., one agent were assigned to perform only one task.

For testing our system with respect to sequential processes, a simple process were defined in which 42 tasks is sequence were put. In the decentralized case, each task was assigned to one single Personal Agent. And in the centralized case, each tasks was assigned to a single Personal Agent. For the centralized case, when processing a single instance, the elapsed time was of 96.275 seconds and for 20 instances the elapsed time was of 1778.313 seconds. For the decentralized case the elapsed time was of 79.811 seconds for 1 process instance and 434.517 seconds

for 20 instances. As can be appreciated in Fig. 4(a) even for 1 process instance, the decentralized approach overcomes the centralized one.

In other experiments, we tested our system with respect to parallel split and synchronization, The process used for this test consisted of a single task (thread) that splits into 40 different thread composed by one task each. Afterwards, all threads converges into another single one. In the centralized case, the duration of one instance execution was 102.411 seconds and for 20 instances it lasted 4068.367 seconds, i.e., over an hour for executing 20 instances. And the decentralized case lasted 47.31 seconds for one instance and 523.287 seconds for executing 20 instances of the same process. Figure 4(b) shows a comparison between these two approaches for workflow process execution. It is clear that a decentralized approach overcomes a centralized one in execution time.

The results presented in this chapter demonstrate that the distributed and decentralized execution of workflow processes outperforms a centralized architecture for the basic workflow patterns. It is clear that these results extrapolate to more complex patterns, which are combinations of the basic ones.

We think these results are clear indication that the decentralized architecture has advantages in terms of scalability, which is a very important issue for large organizations. Indeed, in the experiments we can see that some of the performance curves for the centralized version grew faster than linear.

The reason why the decentralized architecture outperforms the centralized approach, in terms of scalability, is that in the latter we are increasing the number of process instances over one single thread of execution (one Personal Agent), eventually saturating it; this single thread becomes a bottleneck and produces an increasing time overhead. That explains why, in the graphs presented, with an increasing number of instances, time increases not linearly, but worse (we did not investigate whether in the centralized case time was polynomial, exponential or other, but clearly is not linear).

## 6 Related Work

In general, other agent-based workflow architectures [10–16], emphasize the negotiation aspect of multiagent systems and their distributed nature. Thus, they proposed a distributed workflow system as well. However, they centralize the workflow process execution in one single agent (called Workflow Agent or Trigger Agent). In section 5, a comparison between a decentralized process execution and a centralized one was presented. Results demonstrate that a decentralized workflow process execution is better than a centralized one in terms of scalability. In addition, in those architecture there are several agent instantiation at run time under no control. In other words, they assume an environment with unlimited resource while in real environments that cannot be assumed. Our system does not makes that assumption since all agents are predefined to run at system start up. Snd besides, the required quantity of agents in our system is linear to the quantity of workers in the organization.

Compared to agent-enhanced approaches [17–19] our system architecture allows to automate behavior, i.e. agents can execute tasks on its own without human involvement, agents react to its environment, agents can adjust themselves, e.g., they can create new tasks or new routing depending on the circumstances, and finally, agents have high level features such as learning, negotiation, and planning [20]. In other words, an agent-based application has more benefits than an agent-enhanced workflow application since in the agent-enhanced workflow application agents' behavior is limited to the possibilities of the underlying WfMS.

Other architectures have been proposed for distributed workflow engines [21], distributed components of workflow patterns [22], and a distributed architecture in which components get communicated via ontological messages [23]. However, in traditional distributed system, all decisions, coordination and cooperation are hard-coded at design time. Additionally, the elements of these systems share a common goal. These are remarkable differences between this kind of systems and multiagent systems [24] since in the latter, the agents may not share common objectives and therefore they must act strategically, so that they can achieve the outcome they most prefer. In addition, agents are assumed to make decisions about what to do at run time (acting autonomously) while traditional distributed systems cannot.

## 7 Conclusions

We presented in this paper a multiagent-based architecture that supports decentralized workflow processes execution. The proposed solution for this purpose consisted in breaking down the workflow process execution and the process flow control into small execution units handled by distributed agents.

A prototype was developed in order to prove that the proposed solution for decentralized workflow process execution performs better than a centralized approach. Experiments were setup combining some of the workflow patterns and for different number of process instances. The results were, in the two experiments, conclusive since the decentralized approach outperforms the centralized version. These results prove that a decentralized approach for workflow process execution is more scalable than a centralized one.

As future work, we plan to include support for the remaining and more complex of the workflow patterns [9], allow agents to perform automated tasks without human intervention, e.g., when a task requires to incorporate information automaticaly from particular information sources or alarms to be triggering because of something happened in a legacy application.

Also, our decentralized proposal makes it possible to start a process execution and at some part continue it within another organization. This would be inter-organization workflow, which has great economic potential.

## Acknowledgements

This work was supported by the Monterrey Tech's Research Grant CAT011.

## References

1. Koulopoulos, T.M.: The Workflow Imperative. Van Nostrand Rainhold, New York, USA (1995)
2. Simon, A.R., Marion, W.: Workgroup Computing. Workflow, Groupware and Messaging. McGraw-Hill, New york, USA (1996)
3. Aguirre, J., Brena, R., Cantu, F.: Multiagent-based knowledge networks. Expert Systems with Applications **20** (2001) 65–75
4. Brena, R., Aguirre, J.L., Trevino, A.C.: Just-in-time information and knowledge: Agent technology for km bussiness process. In: Proceedings of the 2001 IEEE Conference on Systems, Man and Cybernetics, Tucson, Arizona, Octubre 7-10, IEEE Press (2001)
5. Ceballos, H., Brena, R.: Combining local and global access to ontologies in a multiagent system. Journal of Advanced Computational Intelligence and Intelligent Informatics **9** (2005) 5–12
6. Brena, R., Aguirre, J.L., Treviño, A.C.: Just-in-Time Information and Knowledge: Agent Technology for KM Bussines Process. In: Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference, Tucson, USA (2001)
7. P., C.A.M.: Decentralized Execution of Workflow Processes Using a Multiagent Architecture. Msc. in intelligent systems, Tecnológico de Monterrey, Campus Monterrey, Monterrey, México (2005)
8. Bellifemine, F., Poggi, A., Rimassa, G.: Jade - a fipa-compliant agent framework. In: Proceedings of PAAM99, London. (1999)
9. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow Patters. Technical Report FIT-TR-2002-02, Queensland University of Technology, Brisbane, Australia (2002) http://is.tm.tue.nl/research/patterns/.
10. Chang, J.W., Scott, C.T.: Agent-based Workflow: TRP Support Environment. In: Fifth International World Wide Web Conference, Paris, France (1996)
11. Jennings, N., Faratin, P., Johnson, M., Brien, P., Wiegand, M.: Using Intelligent Agents to Manage Business Processes. In: First International Conference on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM96), London, UK (1996) 345–360
12. Manmin, X., Huaicheng, L.: Cooperative Software Agents for Workflow Management System. In: Fifth Asia-Pacific Conference on Communications and Fourth Optoelectronics and Communications Conference (APCC/OECC'99), Beijin, China (1999) 1063–1067
13. Yunlong, Z., Hongxin, L., Jinsong, X., Hongtao, W.: The Design of Cooperative Workflow Management Model Based on Agent. In: 31st International Conference on Technology of Object-Oriented Language and Systems, Nanjing, China (1999)
14. Gou, H., Huang, B., Liu, W., Ren, S., Li, Y.: An Agent-based Approach for Workflow Management. In: IEEE International Conference on Systems, Man, and Cybernetics, Nashville, USA (2000) 292–297
15. Botha, R.A., Eloff, J.H.P.: Access Control in Document-centric Workflow Systems – An Agent-based Approach. Computers & Security **20** (2001) 525–532

16. Stormer, H.: A Flexible Agent-based Workflow System. In: The 5th International Conference on Autonomous Agents, Montreal, Canada (2001)
17. Odgers, B., Shepherdson, J., Thompson, S.: Distributed Workflow Co-ordination by Proactive Software Agents. In: Intelligent Workflow and Process Management. The New Frontier for AI in Business IJCAI-99 Workshop, Stockholm, Sweden (1999)
18. Dogac, A., Tambag, Y., Tumer, A., Ezbiderli, M., Tatbul, N., Hamali, N., Icdem, C., Beeri, C.: A Workflow System through Cooperating Agents for Control and Document Flow over the Internet. In: CooplS '02: Proceedings of the 7th International Conference on Cooperative Information Systems, London, UK, Springer-Verlag (2000) 138–143
19. Hulaas, J.G., Stormer, H., Schonhoff, M.: ANAISoft: An Agent-based Architecture for Distributed Market-based Workflow Management. In: Software Agents and Workflows for Systems Interoperability workshop of the Sixth International Conference on CSCW in Design, London, Canada (2001)
20. Yan, Y., Maamar, Z., Shen, W.: Integration of Workflow and Agent Technology for Business Process Management. In: The Sixth International Conference on CSCW in Design, London, Canada (2001)
21. Ceri, S., Grefen, P., Sáchez, G.: WIDE - A Distributed Architecture for Workflow Management. In: IEEE 7th International Workshop on Research Issues in Data Engineering (RIDE '97) High Performance Database Management for Large-Scale Applications, Birmingham, UK (1997)
22. Ferreira, J.P., Ferreira, H., Toscano, C.: Distributed Workflow Management Enactment Engine. In: International Conference on Industrial Engineering and Production Management, Porto, Portugal (2003)
23. Blake, M.B.: Agent-Based Communication for Distributed Workflow Management using Jini Technologies. International Journal on Artificial Intelligence Tools **12** (2003)
24. Wooldridge, M.: An Introduction to MultiAgent Systems. John Wiley and sons, LTD, Baffins Lane, England (2001)

# Cooperation in multirobotics environments

Félix Orlando Martínez Ríos[1] and Carlos Rodríguez Lucatero[2]

[1] Universidad Panamericana campus Ciudad de México
Computer Science Department
fmartin@mx.up.mx
[2] Universidad Iberoamericana
Computer Science Department
carlosr.lucatero@uia.mx

**Abstract.** In the present article we presented the results of a simulator in order to evaluate the performance of multiagent systems. We approached the problem of exploration of unknown environments using three types of agents: one with ample observation capacities but without moving ability, others with big displacement capacity but whose observation ability is limited to the recognition of their present position (explorers), and finally another group of agents with possibilities of high displacement and load capacity, and narrow sensorial capacity (shippers).
In this work we also present a proposal about paths memorized by agents, based on the creation of a tree of obstacle-free paths. This tree is stored in a blackboard to which all the shipper agents have access, and enables them to choose the best trajectory from their current position to the point in which the samples have been discovered. This work also displays a strategy of collaboration and conflict resolution based on a contract net-like mechanism.

## 1 Introduction

The problem which we solved with this multiagent system consists on the exploration of an unknown environment [12]. This space is composed by a set of obstacles and samples (objects to be collected) that have to be loaded and bring to a special point which we will call $ET^0$. We will analyze three different approaches to solve this problem:

1. In the first method we have agents who explore and load samples to the point $ET^0$ without collaboration.
2. The second approach besides using previous strategy, incorporates collaboration between agents, such that when an agent discovers samples in the environment, when returning to the point $ET^0$ it leaves landmarks that can be used by itself or other agents to follow this way and then go back to the point where samples were discovered.
3. Our approach is to divide the agents in three different types: the first one with ample calculation and observation abilities ($MR^1$), the second ($MR^2$)

© A. Gelbukh, R. Monroy. (Eds.)
Advances in Artificial Intelligence Theory
Research on Computing Science 16, 2005, pp. 89-98

with great possibilities of displacement and capacities of observation limited to its present position and the third ($MR^3$) with load and displacement possibilities as well as capacities of observation limited to its present position.

Proposals one and two were made by Wooldridge in [12], whereas the third one is our proposal of solution.

### 1.1 Types of obstacles used in the simulations

In our experiments we used randomly generated obstacles as well as obstacles with some kind of symmetry that makes some subregions of the environment become hard to access by the agents.

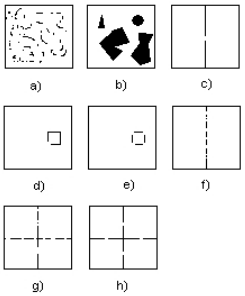



**Fig. 1.** Obstacles used in the simulations

The obstacles of the Figure 1.a) were randomly generated in all the search space . From now on we will call it type 1 obstacle. We will also identify like this type of obstacle those that are constructed by the user, as it is shown in Figure 1.b), guided by a graphical software. In the obstacle of Figure 1.c) the environment is divided vertically. The agents can move from one side to the other through a small hole placed in the middle of the obstacle. This will be identified as type 2 obstacle. Figure 1.d) presents a small box that completely surrounds the point $ET^0$. This box has a hole in the left bottom corner through which the agents can leave and go back to $ET^0$. This obstacle will be identified as type 3. Figure 1.e) displays an obstacle that is similar to the previous one, but in each corner of the box, it has a hole. This will be identified as type 4. The obstacle of type 5, is displayed in Figure 1.f). Like the type 2, has a vertical line that divides the space in two equal sized areas. Unlike the type 2 obstacle this one has several random holes. Figure 1.g) shows type 6 obstacle, made up by two perpendicular lines that divide the search space in four regions of equal dimension. These lines have several random holes that enable the communication between different subregions. Finally, in the Figure 1.h) it can be observed the type 7 obstacle, that like the type 6, has two perpendicular lines but in this case it only has one hole that connect subregions.

## 2 Experimenting with Wooldridge's proposals

The first solution proposed by Wooldridge [12] consisted in a set of robots that do not communicate to each other and which behavior were basically reactive. Robots leave the main ship (in $ET^0$ point) and begins to explore using random movements, when an agent finds samples then load them and return to $ET^0$ following the decreasing gradient field. The other solution given by Wooldridge consisted in a multiagent system with a cooperative behavior (simple but very limited). In that case Wooldridge assumes that the agents return to point $ET^0$ leaving radioactive landmark in the path. Because we couldn't find numerical results of Wooldridge proposals, we have to simulate his models in order to compare these results with our results.

Previously we did some tests to determine the number of runs necessary to obtain average times that not differ from each other more than 5%.

This strategy of collaboration improves a little the first given solution [2, 6, 7], because it leaves at least a sign of the way to follow from the position of the samples to the ship. Unfortunately if an agent passes over the marks they are erased. In addition there is no guarantee that when arriving at the group of samples at the end of the way they remain there. Another inherent problem is that when arriving at an intersection of ways, there is not a criteria to decide which path must be taken.

From results shown in Table 1 we obtain the following conclusions:

1. For the case of obstacle 1 and 4 both models fulfilled the total of the task in 100% of the simulations, but the time for collaboration case was 23% better for obstacle 1 and 13% better for the obstacle 4, than the time taken by the simulator without collaboration.
2. In the case of type 3 obstacle the time improvement was 28% and in the 100% of the simulation cases task was completed.
3. When using type 6 obstacle the percentage of success in the total fulfillment of the task did not improve remarkably, but the total time was improved in a 28%.
4. For the case obstacle 7 the task fulfillment time was improved as well as the percentage of times that the simulator completes the task until a 100%.
5. Concerning the obstacles of type 2 and 5 their total time was not improved but an increase of 32% and 26% was obtained respectively.

## 3 Description of our proposal

Our environment consists of two dimension finite space, that will be represented by a matrix $ET$ composed by $n \times m$ cells.

Each element $ET_{i,j}, 1 \leq i \leq n$ and $1 \leq j \leq m$ they represent only one of the following components: empty space, a robot $mr_k^l$, a number $r \in \mathbb{N}$ of samples, or an obstacle.

The samples located in each grid of the search space, are placed randomly by the simulator.

| | without collaboration | | with collaboration | |
|---|---|---|---|---|
| Obstacle type | Time (s) | success percentage | Time (s) | success percentage |
| Obstacle 1 | 574 | 100 | 445 | 100 |
| Obstacle 2 | 1867 | 54 | 4205 | 86 |
| Obstacle 3 | 1048 | 96 | 761 | 100 |
| Obstacle 4 | 468 | 100 | 411 | 100 |
| Obstacle 5 | 1008 | 87 | 2006 | 100 |
| Obstacle 6 | 994 | 92 | 723 | 93 |
| Obstacle 7 | 2856 | 55 | 2551 | 81 |

**Table 1.** Simulation results of Wooldridge's models with and without collaboration for the seven different obstacle types. We show the total average time to complete the task and the percentage of success in the different simulations

We also have a distinguished element of $ET^0$ with coordinates $i_0$ and $j_0$ that we will defined as starting point and that can be any of the $ET$ cells, with the constrain of not being surrounded by obstacles preventing the access to this point.

We also divided the agents in three classes taking into account: its observation capabilities, processing power [11, 6, 7], displacement abilities and loading capacities [10]. These classes are:

1. Class $MR^1$: To this class belongs just one agent. It has observation, communication, calculation and storage possibilities, but cannot move. Its observation capabilities enable him to determine if an obstacle-free straight path joining two cells $ET_{i,j}$ and $ET_{u,v}$ exists. Similarly it can store the received information of the agents of class $MR^3$ (shippers) concerning the obstacle-free straight paths that have been used to reach some $ET_{i,j}$. It also has ample communication capacities that enable him to communicate, as mediator, with all the remaining agents.
2. Class $MR^2$: Here we will have a set of agents having large displacement and observation capabilities. We will call them explorers. Their processing and storage power are small and its main function is to explore the environment to determine the existence of obstacles and samples. These agents contract the agents of class $MR^3$ who will make the recollection of the samples. We will call them $mr_k^2$ agents.
3. Class $MR^3$: To this last class belong the agents with large loading and displacement capacities but with no observation abilities. These agents will be called shippers and be denoted as $mr_k^3$. These are in charge to collect the samples and bring them to the $ET^0$ point. These agents use for their displacement the obstacle-free segments of the path that already have been discovered by other agents of the same class and which are stored in the $MR^1$ agent of class .

### 3.1 Behavior of the agents

The explorer agents, in class $MR^2$, leave the $ET^0$ point, and move randomly, same as in Wooldrige's model (we focus our attention improving the efficiency of our proposal based in cooperation between agents). If a cell is empty (not occupied by another agent and without obstacles), these agents will move to it. Once in the cell they verify the presence of samples, and if it is the case, then begin the hiring of shipper agents process (belonging to $MR^3$ class).

Let us suppose that the explorer agent arrive at the cell with coordinates $u, v$ in which it discovers samples, then begins a hiring shipper agents process based on the contract network mechanism [2, 6, 5] and using $KQML$ [9, 3, 8, 4] as the communication language. The agents messages are sent to a blackboard where can be read by the rest of the agents on the system. The agent in $MR^1$ is charged to support all the blackboard information. Shipper agents that are not currently engaged in a task can read the blackboard to see if they find there hiring messages.

The explorer agents follow a task allocation rule that tries to reduce the number of agents that participate in the recollection. For that reason, once the blackboard was reviewed (agents are ordered decreasingly by their loading capacity) the shipper agents are selected in that order until the amount of samples detected by the explorer agent can totally be loaded. The idea behind this process is to have the smallest possible number of agents moving in the search space and to minimize conflicts produced by crossing paths.

The shipper agents who have been contracted to recollect the samples follow the next sequence of steps:

1. With the information stored in the $MR^1$ class agent, it determines if between the points of coordinates $u, v$ and $i_0, j_0$ an obstacle-free straight path exists. If it exists then it follows the straight line segment that join them and publishes in the tree of discovered paths.
2. If it does not find a straight path in the previous step, then it begins to consult the information stored in the tree of discovered paths. Whenever it arrives at a node of this tree it follows the same behavior of the step 1, to try to arrive at $u, v$. This process continues until it finds a road. In this case a new obstacle-free segment is added to the tree. Otherwise the tree of discovered paths overflows and the task is rejected.
3. If the number of rejected tasks exceeds a given threshold, then the shipper agents move randomly trying to achieve a point where the recollection task can be continued and apply the step 1. If this process also fails then the task is kept in the blackboard for later accomplishment.

### 3.2 The tree of discovered paths construction

In order to understand how the tree is constructed a hypothetical scene is given as example in Figure 2.

The node labeled by 0 corresponds to $ET^0$, the points labeled by $1, 2, 3, 4, 5$ and 6, correspond to cells in the neighborhood where there is a certain number

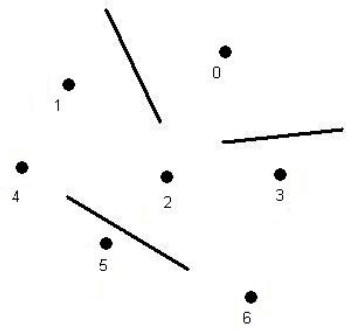



**Fig. 2.** Example of construction of the tree of obstacle-free straight paths

of samples to gather. Finally, the segments of straight lines represent obstacles. Initially the tree is empty. Let us suppose that an explorer agent arrives at point labeled 2, and at this point we have an obstacle-free straight path. Then shippers will arrive at this point and will store a first node in our tree having the coordinates of the achieved point. At this stage the tree is rooted at the 0 node, the coordinates of the point are stored in the node and a descendant node labeled by 2 is added to it (the coordinates of point 2 are stored in the corresponding node). This construction stage is displayed in Figure 3.a).

Later the explorer agents discover points 1 and 3, in this order. Given that they are not reachable by an obstacle-free straight path from 0, then the information of the tree is consulted and it is observed that point 1 can be reached from point 2. This new path is added to the tree as it is shown in the Figure 3.b). Similarly the path to the node 3, from node 2 is added, as it is shown in the Figure 3.c). After that, points $4, 5$ and 6 are discovered, in that order, and added to the tree as it is shown in the Figure 3.c). It is important to notice that point 5 cannot be reached before discovering point 4 or 6. It's clear that this is not a binary tree because more than two paths can be added to the same node.

Many trees can be constructed for the same environment (depending how samples are discovered). This is not and issue because the tree only is useful to access new locations based on previously known locations and not to describe the environment itself.

This mechanism can fail if there is no reachable point from $ET^0$ . In order to avoid this problem the simulator is equipped with a positive integer value representing the maximum number of allowed failures. When this value is reached the shipper agents make a first random walk [1]. After that the initial algorithm is retaken.

### 3.3 Conflict negotiation between agents ready to collect samples

When an explorer agent $mr_m^2$ detects samples, it sends a hiring message to all the shipper agents. This message is attended by all the shipper agents which are

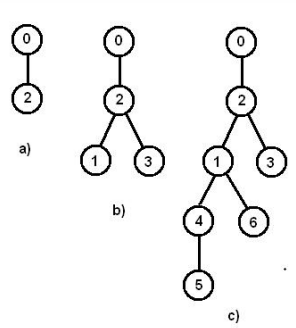



**Fig. 3.** Tree of obstacle-free paths for the example on Figure 2

in the $ET^o$ point. The agents that are already making a recolection will not be able to respond to these messages.

The strategy followed in this negotiation consists on diminishing the number of agents of class $MR^3$ which participate in the sample recollection. This is made by taking the agents that have greater lifting capacity, for which the explorer agent acts like mediator. In this selection process the shipper agents, are sorted in decreasing order of their lifting capacity, and are seleced those with greater capacity until achieving the amount of shipper agents needed to collect the discovered samples.

This strategy has three basic purposes:

- To diminish the number of shipper agents which travel to a point of the environment. Doing this we can guarantee that having less agents we reduce the number of conflicts in crossing paths.
- To maximize the amount of collected samples because the agents are loaded at their full capacity.
- To diminish the amount of information about the environment that must be stored in the free-path tree discovered that is updated by each recollector agent who discovers a new path.

### 3.4 Path conflict resolution by a negotiation mechanism

The negotiation principle followed by the agents in our system tries to optimize the global objective that is to collect the greatest possible number of samples in the smallest period of time. Based on this principle, the negotiation between agents follows the next rules:

1. If an explorer agent $mr_m^2$ and a shipper agent $mr_k^3$ try to occupy the same $ET_{u,v}$ cell, the shipper has occupation priority over the explorer agent.
2. If two shipper agents $mr_m^3$ and $mr_k^3$ try to move to the same $ET_{u,v}$ cell, the shipper agent who is loaded and is going to deliver its load will have occupation priority. The agent who can not occupy the cell, begins a random walk and tries to recover its plan some movements later [1].

3. If two shipper agents $mr_m^3$ and $mr_k^3$ try to move to the same $ET_{u,v}$ cell, and are not loaded then the agent of greater lifting capacity will have ocupation priority over the other. The other enters into a state of random movements for recovering later his original trajectory. In the case that both agents have equal lifting capacity it will be decided randomly who will occupy the cell.
4. If two shipper agents $mr_m^3$ and $mr_k^3$ try to move to the same $ET_{u,v}$ cell, and both are loaded then the agent with the greater possible load will have priority over the other. If both have equal load capacity the decision of who has priority over the other will be at random. The agent whith less priority enters into a random movement state and tries to recover its path after certain number of movements [1].
5. If two explorer agents $mr_m^2$ and $mr_k^2$ try to move to the same $ET_{u,v}$ cell, then it will be decided randomly who will occupy the cell.

### 3.5 Experimental results under our cooperative model

The experiments were made for different proportions of explorer and shipper agents, going from a 10% to a 90% of explorer agents (increasing by 10% steps this amount) and for a 95% of explorer agents. Each one of these proportions was tested with different obstacle types. In Figure 4 we show the average time necessary to complete the 100% of the sample recollection, applied to different obstacle types and for each different explorer and recollector agent proportions. We can draw from Figure 4 the following conclusions:

1. Independently of the obstacle type, it can be observed that the time necessary to complete the task diminishes with the increase on the number of explorer agents until a value of 80% but it starts to increase again from this value which is observed for a 90% and 95%. Evidently more samples are discovered, but there are very few recollector agents to carry out them and these samples are left idle in the blackboard until a new opportunity appears.
2. The best proportion between explorer and recollector agents is between 70% and 80% of explorer agents.

Now we will compare the results obtained with our proposal against the results obtained using the two Wooldrige's models. Analyzing the Figure 5 it can be observed that the average time invested using our proposal to complete at 100% the task, with the different obstacle types, was significantly less than the average time under the Wooldridge's models. Moreover the worst results produced by our proposal (for the case of a 10% of explorer agents) were better than the results obtained using the Wooldridge's proposal.

## 4 Conclusions

1. Our proposal of agents with different capacities concerning observation capabilities as well as loading and displacement abilities, outperform the one that uses only one type of agent proposed by Wooldridge.

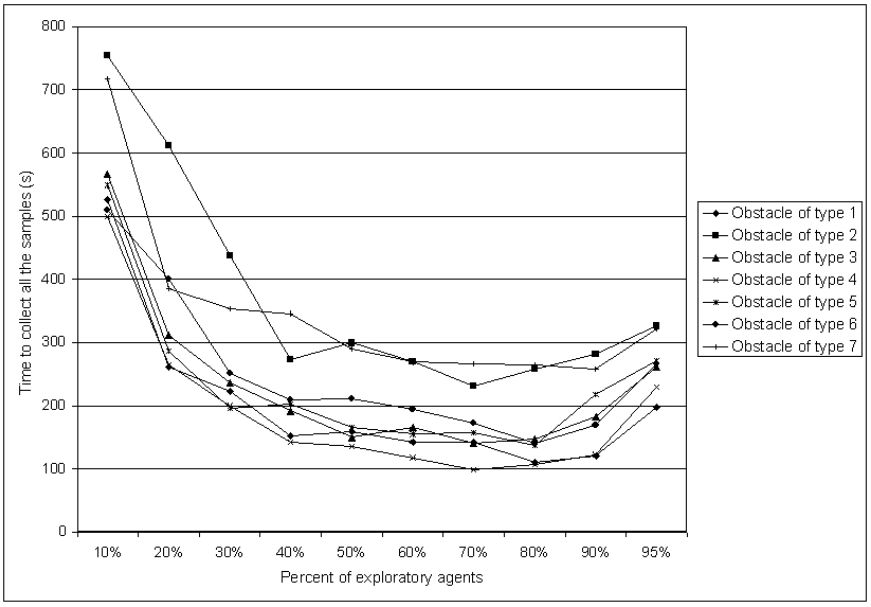



**Fig. 4.** Times to complete the recollection task with different proportions of explorer agents and different obstacle types

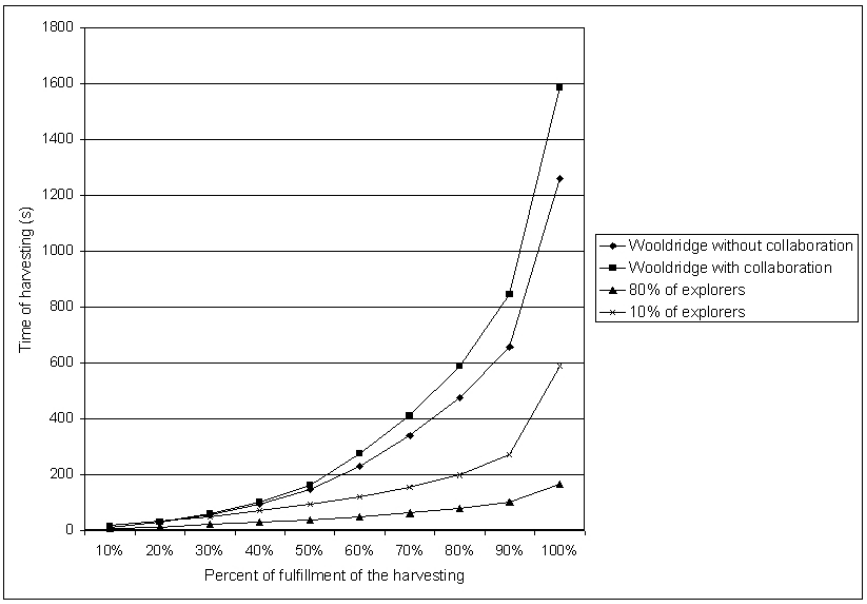



**Fig. 5.** Average time comparison between our model with the Wooldridge model for all the obstacle types

2. The negotiation and collaboration strategy for resolving conflicts, based on giving priority to shippers over the explorers, was quite efficient for these kind of problems.
3. The best performance of the system for the sample recollection, was obtained when using between 70% and 80% of explorer agents.
4. It has been experimentally shown that learning obstacle-free path method used in our simulator is a very efficient recognition form of the search space. In this sense, it must be mentioned that the size of the trees in most of the cases do not exceed a depth of three levels, and because of that the agents have a faster way to reach different points of the explored space.
5. The strategy of random movements of the shipper agents used to solve the problems of unexpected obstacles in their planned trajectories was quite effective, because noncollected samples never appeared.

## References

1. Barraquand, J., Latombe, J.C.: Robot Motion Planning: A distributed representation approach. STAN-CS-89-1257 (1989) Stanford University.
2. Durfee, E.H.: Coordination of Distributed Problem Solvers. Kluwer (1988).
3. Finin, T., McKay, D., Fritzson, R.:An overview of KQML: A Knowledge Query and Manipulation Language. Technical Report (1992) U. of Maryland CS Departemnet.
4. Ginsberg, M.L.: Knowledge interchange format: the KIF of death. AI Magazine archive 12 (1991) 57–63.
5. Haddadi, A.: Communication and Cooperation in Agent Systems: A pragmatic Theory. Springer-Verlag (1996) Heidelberg.
6. Huhns, M.N., Singh, M.P.: Agenst and multiagents systems: Themes, approaches, and challenges. Distributed Artificial Intelligence 1–23 (1998). Morgan Kaufmann San Francisco CA.
7. Jennings, N.R.: Coordination Techniques for distributed Artificial Intelligence, In GMP O'Hare and N.R. Jennings, editors, Foundations of Distributed Artificial Intelligence (1996). 187–210 John Wiley and Sons Inc. New York.
8. Labrou, Y., Finin, T.: A Proposal for a new KQML Specification. http://www.csee.umbc.edu/kqml/papers/kqml97.pdf (1997).
9. Neches, R. Fikes, R., Finin, T. Gruber, T., Patil, R.,Senator, T., Swartout, W.: Enabling Technology for knowledge sharing AI Magazine 12(3) (1999) 36–56 Fall.
10. Rao, A.S., Georgeff, M.P.: An abstract architecture for rational agents. In C. Rich, W. Swartout and B. Nebel, editors, Proceeding of Knowledge Representation and Reasoning, Morgan Kaufmann (1992) 439–449.
11. Shoham, Y.: Agent-Oriented Programming, Artificial Intelligence 60-1 (1993) 51–92.
12. Wooldridge, M.: Intelligent Agents. In G. Weiss editor, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence MIT Press Cambridge MA (1999) 27–77.

# Economics of Cooperation: Social Foraging in Distributed Systems

Ashish Umre[1] and Ian Wakeman[1]

[1] Software Systems Group, Department of Informatics,
University of Sussex, Falmer, Brighton BN1 9QH, United Kingdom
(ashishu, ianw)@sussex.ac.uk

**Abstract.** The sharing and collective processing of information by individuals in any social system is an attempt to reduce the uncertainty associated with key features of their environments by collecting and storing information. By sampling each of its options regularly, an individual gains from being able to exploit them when they are productive and avoid them otherwise. In this way, collection of information can be thought of as a solution to the uncertainty problem that maximises potential opportunities [3], [4]. Some group-living species have evolved effective social mechanisms for reducing uncertainties in their environments. However, doing so may entail certain costs with respect to attributes such as time, energy and attention. In this paper, we explore the cost/benefits of cooperation within the domain of distributed systems, where biologically inspired agents interact with each other using the environment to disseminate information about resources (foraging sites). In the sections that follow, we describe briefly the theory of cooperation, social foraging theory, the simulation model and some experiments to understand/analyse the dynamics of social foraging in stochastic environments.

## 1 Introduction: Social Foraging and Cooperation

To account for the manifest existence of cooperation and related group behaviour, such as Altruism and Restraint in competition, evolutionary theory has acquired two kinds of extension: Genetic kinship theory and reciprocity theory. If the agents are sufficiently closely related, altruism can benefit reproduction of the set, despite loses to the individual altruist. The evolution of the suicidal barbed sting of the honeybee worker could be taken as a paradigm for this line of theory [12].

Many of the benefits sought by living things are disproportionally available to cooperating populations. The problem lies with the fact that while an individual can benefit from mutual cooperation, each can also do so even better by exploiting the cooperative efforts of others. Over a period of time, the same individuals may interact again, allowing for more complex patterns of strategic interactions. [10] Argues that there are at least three ways that cooperation can evolve among unrelated individuals: reciprocity, group selection, and by-product mutualism. Though, kin selection is a fourth candidate.


*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 99-108*

As well as the existence of group, team and partitioned tasks in complex societies, another facet of higher-level functionality is a shift from individual to social/group foraging strategies. [18] identified six foraging strategies in ant colonies: (1) '*individual foraging'* foraging without cooperation and communication with others; (2) '*tandem running'* a scout guides one recruit to the food source with or without trail laying; (3) '*group mass recruitment'* the scout guides a group of recruits to the source, usually laying a trail to the nest; (4) '*mass recruitment'* the scout lays a trail while returning to the nest which guides recruits to the food source; (5) *'trunk trail'* semi-permanent trails guide foragers to long-lasting food sources; and (6) '*group hunting'* a group leaves the nest and forages collectively in a swarm along a well-defined trail system. These strategies also appear to be correlated with a decrease in the autonomy of the individual foragers themselves [19]. That is, there is a shift from information processing by individuals to emergent properties of a set of essentially probabilistically behaving individuals mediated through signals, i.e. a set of trail pheromones. For instance, in an individual foraging strategy the worker must rely on its own information, navigating back to the nest using the sun or other landmarks (e.g. the desert ant *Cataglyphis bicolor*).

In tandem running, a successful returning forager can recruit just one individual and passes on information of where the food source is by physically leading the recruit to the source (e.g. *Leptothorax*). However, with more complex strategies trail pheromones can pass the information not just to one other recruit but to many. There is no need for an individual to be able to navigate back to the nest using the sun or a prominent rock but can simply orient ('smell') their way along a chemical trail (e.g. *Atta*). Despite the apparent simplicity of this task, foragers experience a constant probability per unit distance of losing the trail. Seemingly counterintuitive, this apparently errant behaviour has been shown to be very adaptive at the group-level [20, 21]. Once lost, these workers become scouts who can search for new sites. However, it appears that the error rate is sufficiently tuned so that enough foragers do not lose the trail and thus can exploit the source whilst enough become scouts enabling a constant supply of new sources. (Parallel behaviour is known in honeybee foraging in which the directional information in waggle dances is imprecise) [22]. It seems that the complexity emerges at the level of the trail network (or group), which can adaptively adjust to fluctuating food dispersion or density. Thus, the foragers are a 'group-level adaptive unit' [5, 23], and also see [24].

## 2 Then again, how advantageous cooperation really is?

The acquisition and use of socially acquired information is commonly assumed to be profitable. But, there could be scenarios where the use of such information either provides no benefit or can actually incur a cost. It is suggested [2] that the level of incompatibility between the acquisition of personal and socially acquired information will directly affect the extent of profitability of the information, when these two sources of information cannot be acquired simultaneously, because of cognitive or

physical constraints. Also, a solitary individual's behavioural decisions will be based on cues revealed by its own interactions with the environment.

However, in many cases, for social animals the only socially acquired information available is the behavioural actions of others that expose their decisions, rather than the cues on which the decision was based. In such a situation it is thought that the use of socially acquired information can lead to *information cascades* that sometimes result in sub-optimal behaviour.

In our experiments, we look for results that suggest the presence of information cascades in the context of information sharing in distributed systems. Designing agents that rely both on individual foraging and shared information, or agents that just rely on shared information. Ongoing studies are focused on understanding whether this might happen in a highly dynamic environment; where there are constant changes in the flow of information about resources that undergo frequent updates.

### 2.1 Cost of cooperative efficacy

In any social group, individuals possess various behaviours that define the assortment of the interactions at all sorts of levels, individual, groups, cliques, teams etc. The social foraging theory suggests that, the functional consequence of an individual's foraging behaviour depends on both the individual's own actions and the behaviour of other foragers. There may be conflicts of interest between signallers and receivers. Where such a conflict exists, the receiver's need to acquire information may favour sensitivity to the cues provided by the behaviour and appearance of the signaller. In turn, this sensitivity may give rise to opportunities for manipulation and exploitation by the signaller.

It is understood that exploitative strategies are unlikely to persist in the long run, because they generate selection for a change in receiver responses. However, it is argued, that the evolution of exploitation may prove a recurrent, though, transient phenomenon. There are costs associated with broadcasting information publicly, as exemplified by the production of *"food vocalisations"* in many social animals. The issues that come under this context are, dangers of predation, and mass recruitment to a very less profitable resource may lead to starvation. This is equivalent to the "Slash Dot" effect that the Internet sometimes experiences.

Other costs within the context of a social system are cost of misinformation (lying), cost of accessing/using the resources and cost of signalling/cooperation. We use foraging games to analyse the economics of Kleptoparasitic[1] behaviour, to predict the ecological circumstances under which the behaviour is maintained. Other costs are expressed as survival rate; if an agent keeps failing/delaying to locate resources for

[1] *Kleptoparasitism* refers to all forms of exploitation of others' food discoveries or captures. It constitutes the information-sharing models in the Social Foraging Theory paradigm.

the requested processes/services it gets penalised and if this increases above a threshold, then the agent dies and a new agent replaces the old agent.

## 3 Model Overview

We implement a discrete-event simulation of cooperative (collaborative) agents, which share information (through the environment, Stigmergy[2]) about the location of resources. A process generator ($P$) generates processes/requests/tasks with Poisson distribution. Processes enter the system queue at the start of the simulation, where they wait to be allocated to $N$ agents (which are initialised randomly). An agent gets allocated a process/task. Individual processes/tasks require a certain number of resources/services ($r_1$, $r_2$...$r_n$) that it requires for the successful completion/execution of the process.

The resource generator ($\Re$) generates a random number of resources for the successful execution/completion of a request. When an agent encounters some information about a resource/service, it probabilistically stores the information in its resource vector and/or publishes the information onto a "HotSpot", if it decides to share it with others.

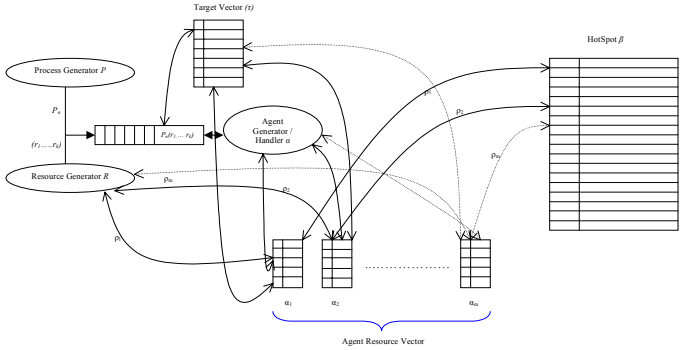



**Fig. 1.** Schematic representation of the information dissemination system.

If the agent encounters a resource it is searching for, it locks the resource, provided it's available at the time and marks the resource entry in the target vector (which contains the list of processes waiting to be finished and the status of the resources) under the specific process. Once all the required resources/services have been located, the process is executed. The agent can only lock the resource for a fixed time after

[2] The term *"Stigmergy"* was first introduced by *Pierre Paul Grassé*, a French entomologist, in 1959. He used the term to describe the coordination of activities of ants in carrying out complex activities, such as nest building or foraging, without direct communication amongst themselves. It is evident that stigmergy describes a form of asynchronous interaction and information interchange between entities mediated by an "active" environment.

which it will have to rejoin the queue. The agent incurs a cost once it has locked a resource. A resource diminishes by a certain value while the process/task is being executed. The *j*th process assigned to the *i*th agent is $p_{ij}$, and costs it $C_{ij}$ . Individual resource cost is $C_{ij}^{r_n}$ , for the resource $r_n$. Cumulative cost associated with *j*th process is,

$$C_{ij} = \sum_{n=1}^{N} C_{ij}^{r_n} \quad (\mathbf{1})$$

Agents can cooperate and form groups to collaboratively execute the process/task or choose to forage alone. The throughput of the system is calculated as a function of successfully completed jobs in the minimum time and with minimal costs. Agents probabilistically ($\rho$) cooperate with other agents, and decide to share information through the HotSpot or not. If the agent incurs a cost which is higher than the cost on its previous task $C_{ij+1} > C_{ij}$ , it then either chooses to collaborate with other agent(s) by forming a group and/or change its degree of cooperation. This acts as a simple adaptive learning mechanism and some form of reciprocity. An agent's cooperative strategy (probability of publishing/sharing information) changes after every process or during successive simulation runs. This is more or less an equivalent NASH equilibrium[3] for the agent.

We have considered the resource handling time as negligible and the process execution time as a random time factor. Other agents looking for the same resource can access the HotSpot and search through the advertised resources/services. The HotSpot contains the information about resources and their location. Each resource published at the HotSpot has a reinforcement value (similar to pheromone deposit) associated with it, which signifies the demand ($\Delta$) of the resource.

Every time an agent accesses resource information at the HotSpot, it reinforces the pheromone deposit so that the resource path continues to exist, whereas if the reinforcement value goes below a certain value, it gets over written by the first new resource that appears in the system. Hence, the table is constantly updated with the latest information about resource paths. Agents attempt to optimise costs locally and globally in accordance with the dynamics of their interactions.

[3] *Nash Equilibrium is a combination of strategies* for the players of a game, such that each player's strategy is a *best response* to the other players' strategies. A *best response* is a strategy, which maximises a player's expected payoff against a fixed combination of strategies played by the others.

### 3.1 Results and Analysis

We analyse some aspects of artificial and biological social systems, such as, optimal number of agents in the system [11], throughput of the system, degree of cooperation (which can depend on an implicit factor of relatedness). Demonstration of the use of Nash equilibrium, to show the "tragedy of the commons" for certain situations both in the simulations and in real life, e.g. Slash Dot effect. How a certain resource gets over exploited because of it being over publicised and may lead to its exhaustion/starvation. Similarities with Caraco's food calling game [1], [15], when agents individually look for resources and on finding it, decide to publish it or not. According to Caraco's model if they decide against publishing the information, then they are more susceptible to predation.

#### 3.1.1 Optimal Group Size?

In general, we observe a peaked fitness function [6] when we analyse the system as a collection of agents trying to maximise the throughput and minimise the delay in acquiring information. The peaked function we see in Fig. 2 illustrates the existence of only one optimal agent population size for which, the throughput of the system is maximum, given that certain other parameters in the simulation remain fixed, like the number of resources.

This suggests that initially an increase in the agent population is beneficial in obtaining a good throughput, but the throughput peaks at some point for a certain size of population implying that there are enough agents to process requests for resources any further increase will result in delays due to queuing for resources. The Increasing Fitness plot is an indication of abundance of resources.

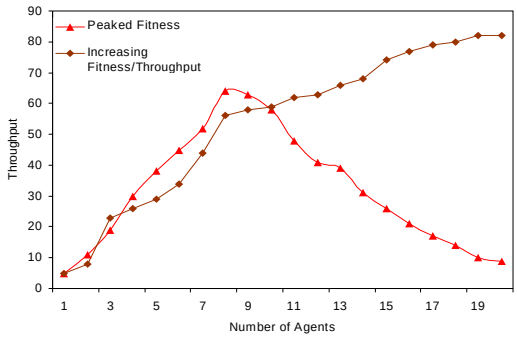



**Fig. 2.** Optimal Number of Agents.

### 3.1.2 Throughput of the System

The time taken to find all the resources for a request can vary depending on the number of resources required. Therefore, we calculated the average time ($\tau_{avg}$) taken for finding the various resources over a series of runs and accumulated the data for all the possible number of resources in the system. We were interested in finding out the trend that follows in terms of time/hops taken to locate all those resources. As seen from Fig. 3(a) out that there is an increasing trend with respect to the number of hops. As the number of required resources increases it takes more time to find them, but the trend shows that there could be a decrease later on in the system as the agents develop an optimum response for each request, as the number of resources increase. This also may lead to a drop in the number of cooperators, meaning that individual foraging can sometimes also be a useful strategy Fig. 3(b). Fig. 3(c) shows the average cost incurred by agents over successive simulation runs. The drop in average cost suggests an increase in information sharing and level of cooperation.

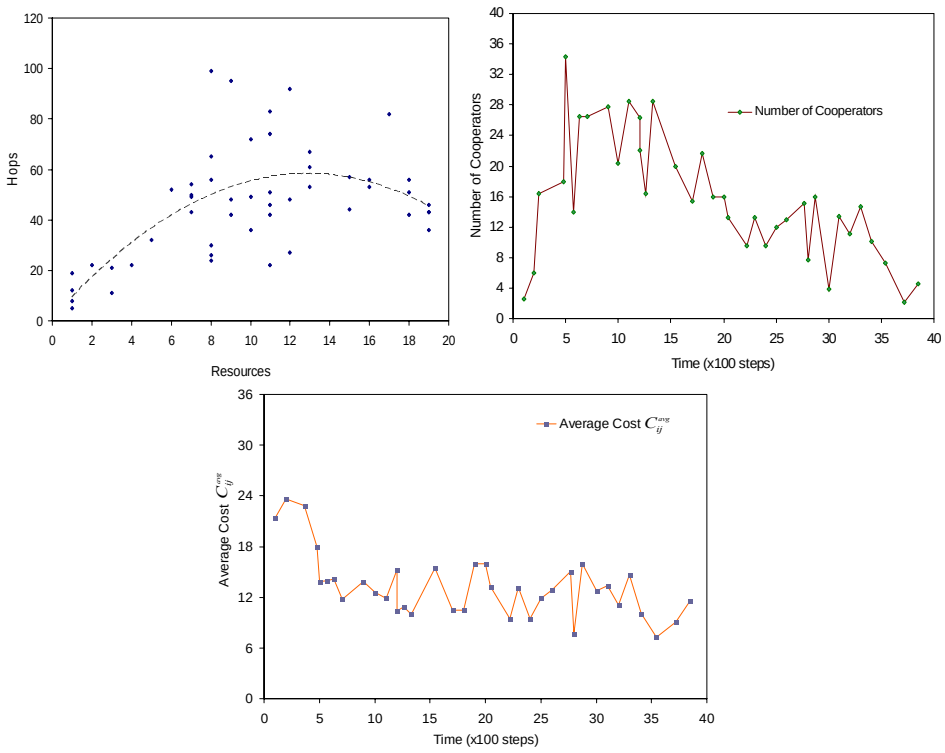



**Fig. 3. (a)** Time ($\tau_{avg}$) to establish a resource path. **(b)** Number of Cooperators. **(c)** Average cost $C_{ij}^{avg}$ over successive runs.

### 3.1.3 Slash dot effect/Kleptoparasitic Behaviour

Slash dot effect, whereby popular data becomes less accessible because of the load of the requests on a central server. The following Fig. 4 demonstrates the percentage increase in the number of agents in the queue for a resource e.g. resource $r_{12}$ in this figure. The figure also displays the corresponding decline in the throughput for processes requiring the service $r_{12}$.

This implies that popular request for a service can lead to it being highly advertised or "vocalised", resulting in the depletion and decreased performance of the service. Therefore, unless there is a way to adapt to this phenomenon, the services will continue to fail or perform at a sub-optimal behaviour. Current work is aimed at studying the possibility of introducing service replication in the locality of the current service. This will distribute the load of the service and help process more requests. Also, it will handle to a certain extent the dynamic nature of the system wherein the services can fail.

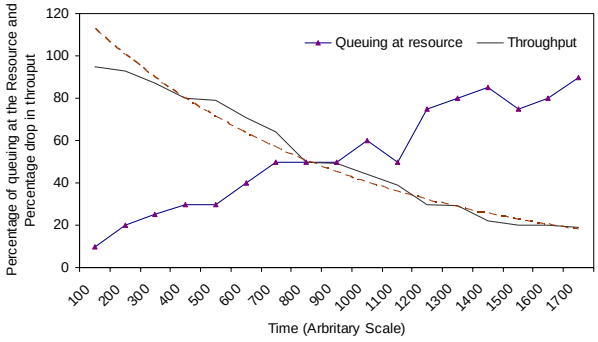



**Fig. 4.** Demonstration of Slash dot effect at a resource $r_{12}$ and the corresponding drop in throughput for processes requiring that resource.

Kleptoparasitic behaviour [14] is observed when an agent frequently refers to the environment for information regarding resources instead of foraging itself. Also, there isn't a change observed in its cooperative strategy, if anything, there is evidence of decreasing cooperation. Implying that the agent is satisfied getting most of its information from other agents that have published/shared the information and itself does not gather information.

### 3.1.4 Vocalisation/Persistence of Resources:

There are various resources that appear and disappear in the system over the duration of the simulation. The requests and usage of resources helps reinforce their life in the simulation. The Fig. 5 below shows a graph indicating the appearance and persistence of resources during one run of the simulation.

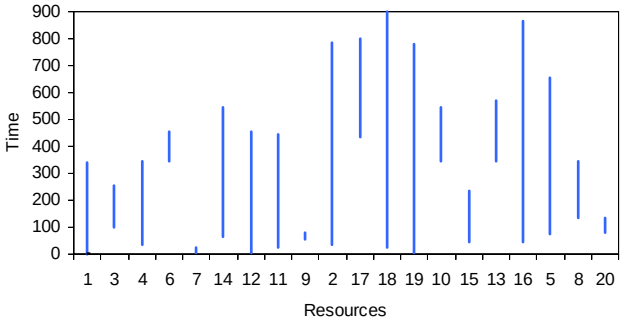



**Fig. 5.** Vocalisation/Persistence of resources.

After the emergence of a resource, its life depends upon the reinforcement or failure. The figure gives an indication of the existence and use of resources at particular times, and helps hypothesize the prospects of their consumption, which is very helpful in evaluating the various strategies being used for sharing information. The perseverance of some particular strategies in the system gives an indication of the behaviour of agents to particular situations and also, determines if certain behaviours recur in agents over time, but without them having the benefit of hindsight.

## 4 Discussion/Conclusions

Our experiments explore various cooperative/competitive strategies that encompass most aspects of social behaviour. Mixed strategy models [8], [9] showing the possibility of freeloaders or lying. Ongoing implementations include scenarios like modelling trust in the system, altruism, and misinformation/malicious agents. To show how information sharing models can make novel, quantitative, and testable predictions concerning social foraging theory, within the application domain of distributed systems e.g. P2P networks.

The experiments reveal some interesting dynamics of the system with respect to the information dissemination algorithm. Our main objective has been to keep the agent imperceptible and its behaviour very simple, and to understand the local dynamics of interacting agents that lead to complex global behaviours. We draw our inspiration for this work from biological social networks, e.g. Ant colonies, Bee colonies, and other relevant theories in behavioural ecology. We are currently developing formalisations for the current algorithmic approach, so as to do a detailed mathematical analysis of the underlying theory. Our study hopefully gives insights into certain kinds of behaviour persistent in the system, which bear some resemblance to biological social systems. Especially to areas such as foraging, danger of predation, sharing information regarding food/nest sites etc, [17], [5], and [16]. Issue of trust and reputation once incorporated into the simulation should yield some more interesting dynamics. The simulation model discussed should eventually be able to help understand some of the contexts in which cooperation emerges, is beneficial or not, and to what extent.

## References

1. Caraco and Giraldeau: Social Foraging Theory, Princeton University Press. (2000)
2. Luc-Alain Giraldeau, Thomas Valone et. al.: Potential disadvantages of using socially acquired information, Phil. Trans. of the Royal Society, London. (2002)
3. Mangel, M.: Dynamic Information in uncertain and changing worlds. J. Theoretical Biology (1990) 146: 317-332
4. Stephens, D.W.: Variance and the value of information. American Naturalist (1989) 134: 128-140
5. Seeley, T.: The Wisdom of the Hive – The Social Physiology of Honey Bee Colonies, Harvard University Press, Cambridge (1995)
6. Clark, C.W., and Mangel, M.: The Evolutionary Advantages of Group Foraging. Theoretical Population Biology, Vol. 30 (1986)
8. Axelrod: The Evolution of Cooperation, Basic Books Inc. (1984)
9. Axelrod: The Complexity of Cooperation, Agent-Based Models of Competition and Collaboration. Princeton University Press. (1997)
10. Dugatkin: Game Theory and Animal Behaviour, Oxford University Press (1998)
11. Pacala, S.W., Gordon, D.M., Godfray, H.C.J.: Effects of social group size on information transfer and task allocation. Evolutionary Ecology, (1996) 10, 127-165
12. Hamilton: Altruism and Related Phenomena, Mainly in Social Insects. Annual Review of Ecology and Systemics (1972) 3:193-232
13. Dall and Johnstone: Managing Uncertainty: Information and insurance under the risk of starvation. Phil. Trans. Royal Society. London. (2002)
14. Hamilton: Kleptoparasitism and the distribution of unequal competitors. Behavioural Ecology. Vol. 13, (2001) 2: 260-267
15. Real and Caraco: Risk and Foraging in Stochastic Environments. Annual Review Ecol. Syst. (1986) 17:371-90
16. Caraco, T., Uetz, G.W., Gillespie, R.G., Giraldeau, Luc-Alain.: Resource Consumption Variance within and among individual: On coloniality in Spiders. Ecology, (1966) 76(1), pp. 196-205
17. Franks, N.R., Pratt, S.C., Mallon, E.A., Britton, N.F., Sumpter, D.: Information flow, opinion polling and collective intelligence in house-hunting social insects. Phil. Trans. Royal Society. London (2002)
18. Beckers, R., Goss, S., Deneubourg, J.L. & Pasteels, J.M.. Colony size, communication and ant foraging strategy. Psyche 96, 239-256 (1989).
19. Jaffe, K. & Hebling-Beraldo, M. J. Respirometry and the evolution of order: negentropy criteria applied to the evolution of ants. In Proceedings of the 11th Conference of the International Union for the Study of Social Insects (Bangalore, India) (ed. G. K. Vereesh, B. Mallik and C. A. Viraktamath), p. 538. Oxford and IPH Publishing Co., New Delhi (1990).
20. Deneubourg, J. L., Pasteels, J.M. & Verhaeghe, J.C. Probabilistic behaviour in ants: a strategy of errors? Journal of Theoretical Biology 105, 259-271 (1983).
21. Deneubourg, J. L., Aron, S., Goss, S. & Pasteels, J.M. Error, communication and learning in ant societies. European Journal of Operations Research 30,168-172 (1987).
22. Weidenmuller, A. & Seeley, T. D. Imprecision in waggle dances of the honeybee (Apis mellifera) for nearby food sources: error or adaptation. Behavioral Ecology and Sociobiology 46, 190-199 (1999).
23. Seeley, T. D. Honey bee colonies are group-level adaptive units. American Naturalist 150, 22-41 (1997).
24. Bonabeau, E. Social insect colonies as complex adaptive systems. Ecosystems 1, 437-443 (1998).

# Computer Vision and Pattern Recognition

# Some Experiments on Corner Tracking for Robotic Tasks

Victor Ayala-Ramirez, Cruz A. Longoria-Mendez, and Raul E. Sanchez-Yanez
E-mail:{ayalav, sanchezy}@salamanca.ugto.mx

Universidad de Guanajuato FIMEE
Tampico 912
36730 Salamanca, Gto.
Mexico

**Abstract.** In this paper, we present a comparison of the performance of Harris and SUSAN corner detection applied to corner tracking tasks in robotic vision. We have tested some corner refining algorithms on both methods and measured their performance when we applied to real images of a real-time sequence. We conclude that for the Harris method, a correlation step using an ideal corner model can improve stability in corner detection. In the other hand, it is better to use SUSAN algorithm without using the correlation step because it degrades its performance. We show also successful applications running at about 8 Hz for both corner detection methods.

## 1 Introduction

Current approaches for visual feature tracking include corners, blobs and edges [1]. Nevertheless, there are some unsolved problems in tracking systems; for example, complex scenes, occlusion problems, moving objects, highlights and reflections, significant illumination changes and motion blur.

Interest points or salient points are points that possess unique properties in an image. Salient features can describe unique objects in an image. One of the most often used features to describe salient point is cornerness property. A corner is a point with a high curvature in the intensity space that can be detected from the discontinuities on the neighborhood of a pixel.

Corner tracking has been used for many applications as diverse as robot visual localization [1], robot homing tasks using omni-directional vision [2], human-computer interfaces for augmented reality [3][4], scene modelling [5] or traffic detection [6].

However, its use in outdoor environment has not been intensively tested. Work to find optimal parameters and performance evaluation of corner tracking algorithms remain to be done. Our problem is to determine which algorithm implement in a robotic corner tracking system for indoor and outdoor environments. Our main application is directed toward characterizing landmarks robustly along an image sequence acquired by a mobile robot during the execution of a navigation task.

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 111-120*

We have compared the Harris and SUSAN corner detection algorithms implemented with some minor refinements. We present the details of our experiments later in this paper. We have proposed two tests to measure performance of corner detection algorithms: i)evaluation of corner detection algorithms on benchmark images and, ii) a stability test. First test is used to find optimal tuning parameters for the two corner detection algorithms compared in this paper. The second test proves stability of corner detection when illumination changes are significant. We also present two sequences where Harris and SUSAN algorithms perform well in complex environments.

## 2 Problem formulation

### 2.1 Harris corner detection

Harris corner detection algorithm was originally developed for robotic applications[7]. Its goal was to match corner points in stereo image pairs to enable a 3D reconstruction of the environment. Its work was an improvement of the work by Moravec [8], who has noted that the difference in intensities of adjacent pixels in edges and uniform regions of an image are small, but at corners the same difference is significantly high in all directions.

Computation of the cornerness property in this method is carried out by convolving a Gaussian mask with the Hessian matrix $H$ of the intensity function of the image and analyzing the resulting matrix $M$.

$$M = e^{-\frac{u^2+v^2}{2\sigma^2}} \otimes H = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} \tag{1}$$

with $\otimes$ a convolution operator.

Cornerness *R(x,y)* of a point $(x, y)$ is then computed as follows:

$$R(x, y) = det(M) - k \cdot (trace(M))^2 = \alpha\beta - k(\alpha + \beta)^2 \tag{2}$$

Interpretation of *R(x,y)* can be related to the behavior of $\alpha$ and $\beta$ as follows:

- When $\alpha$ and $\beta$ are small, we are in an uniform region.
- if $\alpha > 0$ and $\beta = 0$, the point is an edge.
- if both, $\alpha$ and $\beta$, are positive numbers, we have found a corner.

### 2.2 SUSAN corner detection

SUSAN [9] is a corner detection algorithm based in the analysis of the gradient direction of the intensity in a neighborhood around a point. SUSAN stands for the Smallest Univalue Segment Assimilating Nucleus. The principle of this corner detector is to count all the pixels in a circular neighborhood that have an intensity level similar to the central pixel after smoothing with a Gaussian kernel. This region is named the USAN (Univalue Segment Assimilating Nucleus). When the USAN is composed of all the pixels in the vicinity, the region is uniform.

If the USAN is composed of about 50 % of the total pixels, we are in an edge point. A corner point is present when the USAN only covers less than 25% of the neighborhood.

### 2.3 Quality requirements for corner detection algorithms

Main requirements for a corner detection algorithm are [10]:

1. All the true corners should be detected.
2. No false corners should be detected.
3. Corner points should be well localized.
4. Corner detector should be robust with respect to noise.
5. Corner detector should be efficient.

Aspects 1 and 2 are evaluated by testing our implementations using widely used benchmark test images (Figure 1). Evaluation of points 3 and 4 is done by performing a stability test for a corner in an image sequence. This sequence presents a quasi-static image perturbed by illumination noise. Point 5 can be satisfied by achieving a real-time frame rate for the corner tracking system.

## 3 Tests and Results

### 3.1 Parameter tuning for Harris and SUSAN methods.

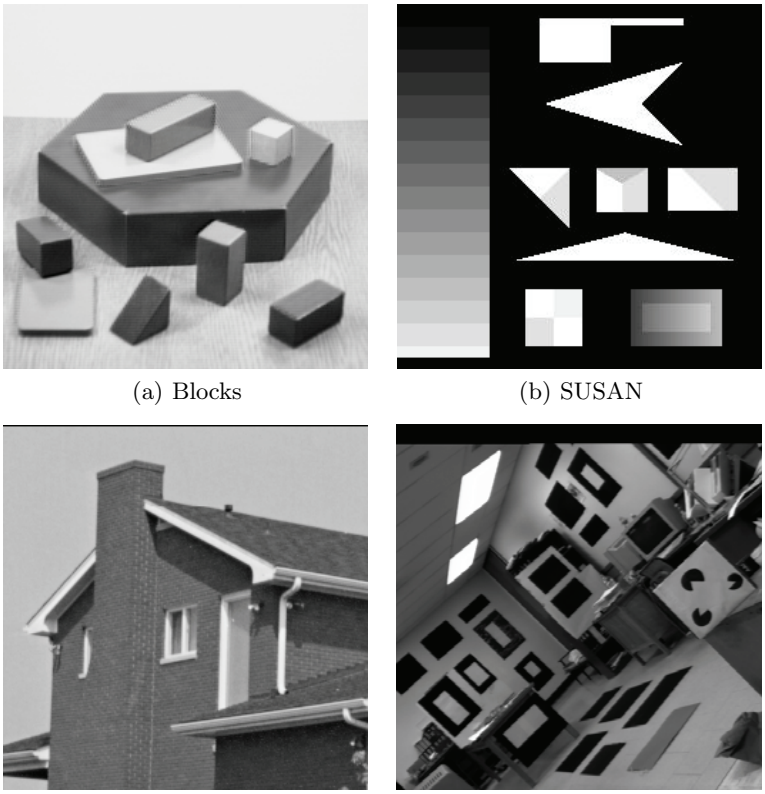
Harris corner detection method is tuned by choosing a variance $\sigma$ for the Gaussian kernel to be convolved with the intensity Hessian matrix. Best results for the variance parameter of Harris detector when applied to benchmark test images are shown in Table 1.

SUSAN method for corner detection is tuned by adjusting the similarity threshold parameter. This parameter controls the area of the pixels belonging the USAN. Best results for the threshold parameter are also shown in Table 1.

For both methods, a different parameter value is needed for each image. This value is selected by choosing the optimal value of the parameter in order to detect all the corners present in the image. Given the different strengths of the corners, this results in some false corners being detected.

### 3.2 Test Protocol.

**Comparison of corner detectors response to benchmark images** Table 1 summarizes the results of the best responses to Harris and SUSAN corner detectors. First column shows to which image (see Figure 1) the detector is applied. Second column shows the actual parameter values used to obtain optimal response. Third column presents the number of corners when the raw algorithm is applies, i.e., no post-processing steps are performed. The results obtained when local minima suppression and thresholding step are shown in fourth column. Graphical results for the SUSAN image when the Harris corner detection



(a) Blocks (b) SUSAN

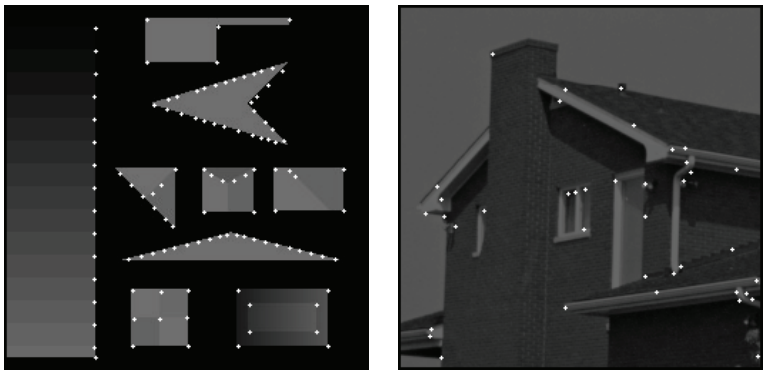
(c) House (d) Lab

**Fig. 1.** Test images used as benchmarks to tune parameters of the compared corner detection methods.

method is used and House image processed by a SUSAN corner detector are shown in Figure 2. Both images were processed using the optimal parameters shown in Table 1.

We can see that SUSAN detector results in a fewer number of corner points than the Harris method for all the images. However we have also found that Harris works better when corner points come from smoother shapes.

**Corner stability for image sequences** In the corner stability test, we have applied the corner detection algorithms to a sequence of 1000 frames of a scene with a fixed target. We have recorded the actual coordinate for the corner, and we have recorded also the temporal evolution of the position of the detected corner. We have tested four conditions for both methods: i) without using a correlation step and non controlled illumination, ii) using a correlation step and non controlled illumination, iii) without using a correlation step and non



(a) SUSAN image Harris detector $\sigma = 1.0$

(b) House image SUSAN detector $t = 25$

**Fig. 2.** Corner detection response for some benchmark test images using optimal parameters.

| Image | Detector | Raw Resp. | Refined Resp. |
|---|---|---|---|
| Blocks | Harris, $\sigma = 1.0$ | 168 | 111 |
| | SUSAN 36 pixels, t=25.0 | 65 | 23 |
| SUSAN | Harris, $\sigma = 0.56$ | 168 | 111 |
| | SUSAN 36 pixels, t=25.0 | 101 | 36 |
| House | Harris, $\sigma = 1.0$ | 143 | 115 |
| | SUSAN 36 pixels, t=25.0 | 28 | 19 |
| Lab | Harris, $\sigma = 1.0$ | 956 | 802 |
| | SUSAN 36 pixels, t=25.0 | 268 | 145 |

**Table 1.** Summary of best responses of detectors when applied to benchmark test images.

controlled illumination plus an illumination perturbation, iv) using a correlation step and non controlled illumination plus a perturbation. We summarize the results of these tests in Table 2. For the sake of space we show only the Gaussian fitting of the corner localization error and the temporal evolution for the cases iii) and iv) in Figures 3 and 4 respectively.

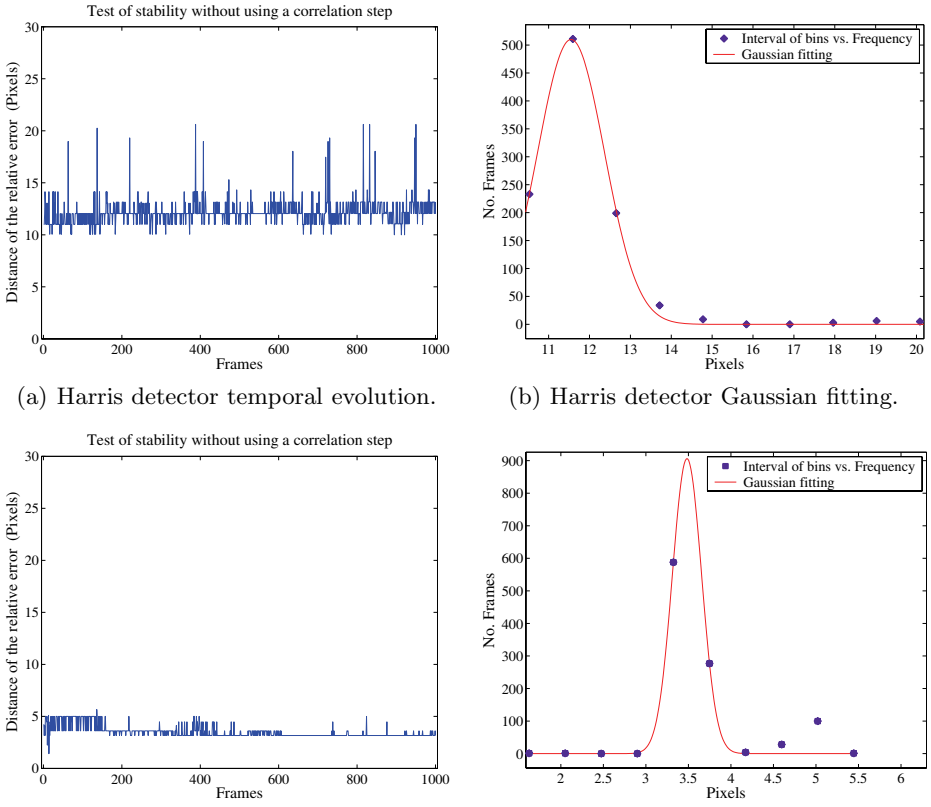
The Gaussian fitting parameters were obtained using the *cftool* provided by Matlab software. Gaussian fitting is of the form:

$$f(x) = a_1 \cdot e^{-\left(\frac{x-\mu}{\sigma}\right)^2} \tag{3}$$

We include also in Table 2 the minimal and maximal errors in corner localization error and the frequency of occurrence along the sequence.

| | | Gaussian fitting | | | Minimal error | | Maximal error | |
|---|---|---|---|---|---|---|---|---|
| Detector | case | $\mu$ | $\sigma$ | $a_1$ | Pixels | Frames | Pixels | Frames |
| Harris | i | 10.17 | 1.321 | 516 | 10 | 520 | 20 | 5 |
| | ii | 7.45 | 0.678 | 1106 | 7 | 900 | 18 | 1 |
| | iii | 11.55 | 1.153 | 510 | 12 | 550 | 20 | 5 |
| | iv | 3.76 | 1.188 | 835 | 3 | 200 | 8 | 30 |
| SUSAN | i | 5.06 | 0.044 | 995 | 5 | 996 | 6 | 1 |
| | ii | 5.36 | 0.867 | 823 | 5 | 850 | 14 | 10 |
| | iii | 3.48 | 0.242 | 906 | 3 | 900 | 5 | 100 |
| | iii | 8.79 | 1.085 | 708 | 4 | 120 | 10 | 550 |

**Table 2.** Summary of results for the corner stability test.



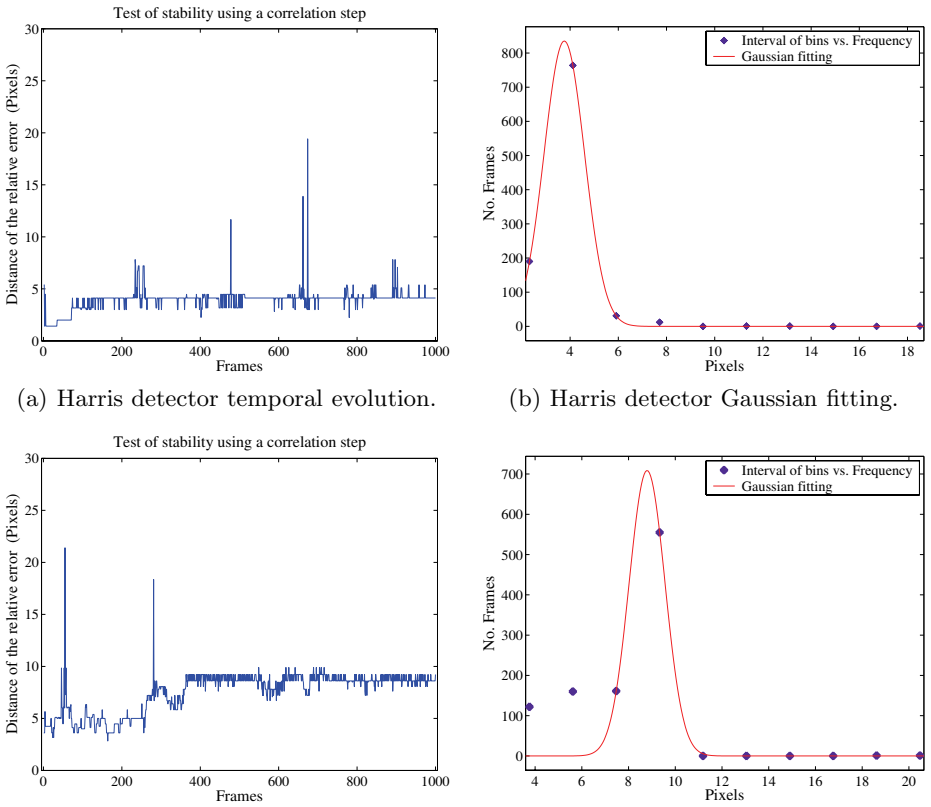

(a) Harris detector temporal evolution.

(b) Harris detector Gaussian fitting.

(c) SUSAN algorithm temporal evolution.

(d) SUSAN algorithm Gaussian fitting

**Fig. 3.** Corner stability test case iii.




(a) Harris detector temporal evolution. (b) Harris detector Gaussian fitting.

(c) SUSAN algorithm temporal evolution. (d) SUSAN algorithm Gaussian fitting

**Fig. 4.** Corner stability test case iv.

### 3.3 Applications

We present in figures 5 and 6, two examples of successful corner tracking applications. Figure 5 presents the tracking of the point of a leaf in an outdoor environment. As we found in previous section, SUSAN method is more robust to illumination changes and in fact, we have obtained better performance with it for this sequence. Figure 6 presents the tracking of the more salient point in a ball that is moved over a textured floor. For this setup, Harris corner detection method has performed better. For both tests, maximal operating frequency was about 8 Hz using a Pentium IV machine running at 2.41 GHz and using 512 MB of RAM.

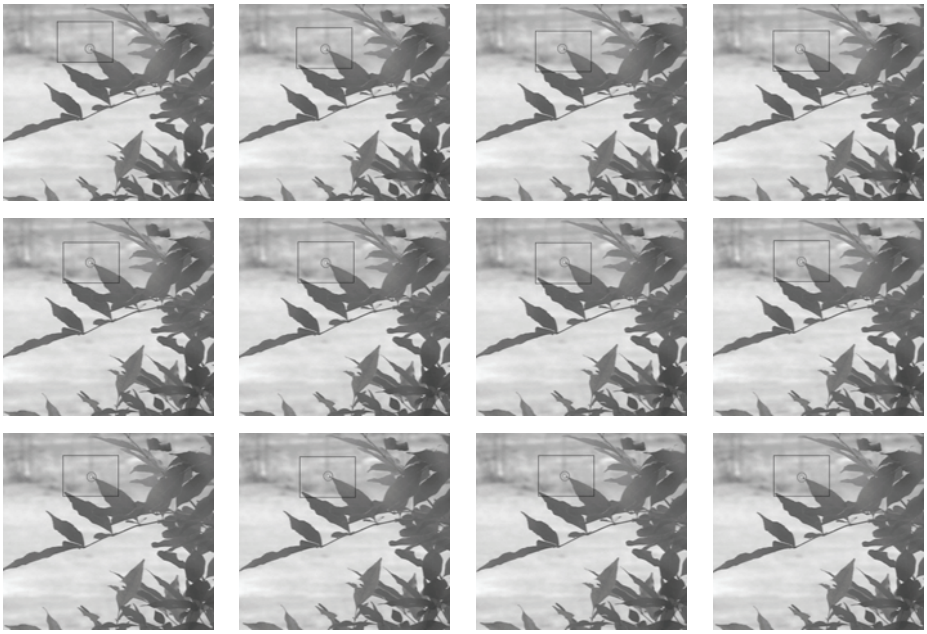


**Fig. 5.** Some frames of the tracking of the maximal cornerness point on an outdoor image sequence using SUSAN method.

## 4 Conclusions and Perspectives

We have presented two experiments to evaluate the performance of Harris and SUSAN corner detection algorithms. We have found that SUSAN algorithm yields better results when the scene includes structured objects. Harris corner detector performs better for scenes containing unstructured objects. Nevertheless, SUSAN algorithm has an error under 12 pixels for a corner stability test under varying illumination conditions.

We will work toward inclusion of this tracking module in a robotic platform. More test will be carried out but using images acquired from the robotic vision system. We will also explore its use for the 3D reconstruction of a panoramic stereo vision system.

## Acknowledgements

This work has been partially funded by the LAFMI project “Concepción de funciones de percepción y planificación para la navegación topológica de un robot móvil en ambiente semi-estructurado interior o natural” and by the UG-DINPO project “Funcionalidades visuales para la navegación de robots móviles”.

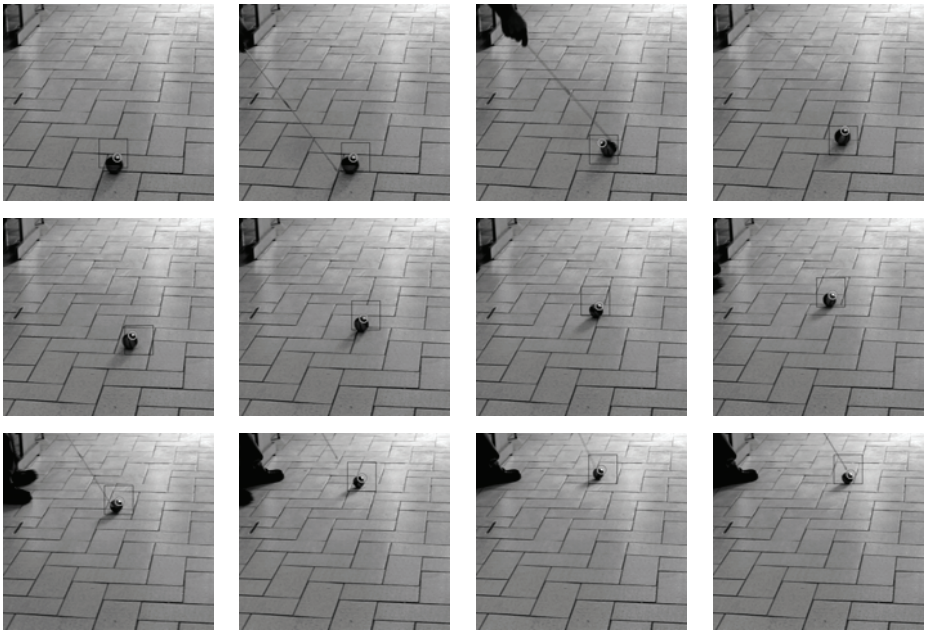


**Fig. 6.** Some frames of the tracking of the maximal saliency point of an object in a complex indoor environment using Harris corner detection method.

## References


1. Brandner, M., Ribo, M., Pinz, A.: State of the art of vision-based self-navigation. In: Proc. 1st Int. Workshop on Robotic Sensing ROSE'03. (2003)
2. Argyros, A.A., Bekris, K.E., Orphanoudakis, S.C.: Robot homing based on corner tracking in a sequence of panoramic images. In: Proc. of the IEEE CSC on Computer Vision and pattern Recognition (CVPR'01). Volume 2., IEEE, Computer Society Press (2001) 3–10
3. Malik, S., McDonald, C., Roth, G.: Hand tracking for interactive pattern-based augmented reality. In: Proc. Int. Symp. on Mixed and Augmented Reality (ISMAR'02). (2002) 117–126
4. Najafi, H., Klinker, G.: Model-based tracking with stereovision for ar. In: Proc. Int. Symp. on Mixed and Augmented Reality (ISMAR'03). (2003) 313–314
5. Skrypnyk, I., Lowe, D.G.: Scene modelling, recognition and tracking with invariant image features. In: Proc. Int. Symp. on Mixed and Augmented Reality (ISMAR'04). (2004) 110–119
6. Beymer, D., McLauchlan, P., Coifman, B., Malik, J.: A real-time computer vision system for measuring traffic parameters. In: Proc. of the 1997 Conf. on Computer Vision and pattern Recognition (CVPR'97). (1997) 495–501
7. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proc. Fourth Alvey Vision Conference. (1988) 147–151
8. Moravec, H.: Towards automatic visual avoidance. In: Proc. Int. Joint Conf. on Artificial Intelligence. (1977) 584

9. Smith, S.M., Brady, J.M.: SUSAN - a new approach to low level image processing. Technical Report TR95SMS1, Defence Research Agency, Franborough, England (1994)
10. Mokhtarian, F., Suomela, R.: Robust image corner detection through curvature scale space. IEEE Trans. on Pattern Analysis and machine Intelligence **20** (1998) 1376–1381

# Pattern Decomposition and Associative Processing Applied to Object Identification

Benjamín Cruz, Humberto Sossa and Ricardo Barrón
Centro de Investigación en Computación – IPN
Av. Juan de dios Batís, Esq. Miguel Othón de Mendizábal
Ciudad de México, 07738, México
E-mails: benjamincruz@sagitario.cic.ipn.mx, hsossa@cic.ipn.mx, rbarron@cic.ipn.mx

**Abstract.** Pattern identification in the presence of noise is a main problem in pattern recognition. An essential characteristic of the noise acting on a pattern is its local nature. If a pattern is thus separated into enough sub-patterns, only few of them will be somehow affected, others will remain intact. In this note we propose a simple methodology that takes into account this property. A pattern is identified if enough of its sub-patterns are also identified. Since several patterns can share some of the sub-patterns, final decision is accomplished by means of a voting mechanism. Before deciding if a sub-pattern belongs to a pattern, sub-pattern identification in the presence of noise is done by an associative memory. Numerical and real examples are given to show the effectiveness of the proposal.

## 1 Introduction

A main problem in pattern recognition is pattern identification in the presence of noise. In real situations usually patterns appear distorted by noise and must be identified despite of this. One approach usually used to identify a pattern from a distorted version of it, is by means of an associative memory by which we reconstruct the distorted pattern. Associative memories have been used for pattern recovering for many years [1-13]. Usually, complete unaltered patterns are first used to build a chosen memory model. Trained memory models are next used to recover a given pattern, given a possibly distorted version it. This allows pattern identification.

One main feature of the noise affecting a pattern is its locality, i.e. the pattern is affected somehow at specific parts or locations; other parts remain unchanged. In this paper we take advantage of this situation and exploit it in two ways. In the one hand, we decompose the pattern into a set of sub-patterns. In the other hand, we make use of an associative memory specially designed to filter the noise affecting the patterns' sub-patterns. The resulting sets of sub-patterns are first used to build a bank of associative memories. During pattern recall a possibly distorted version of a pattern is first decomposed into its sub-patterns. Each sub-pattern is presented to its corresponding memory for noise cleaning. The cleaned sub-pattern is then used to index into a table for the set patterns sharing it. A simple but efficient voting mechanism allows to finally deciding the index of the corresponding pattern.

Lots of models of associative memories have been emerged in the last 40 years, starting with the *Lermatrix* of Steinbuch [1], then the *Linear Associator of* Anderson [2] and Kohonen [3], and the well-known model proposed by Hopfield in 1982, the *Hopfield Memory* [5]. For their operation, all of these models use the same algebraic structure. In the 90's appeared the so-called Morphological Associative Memories

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 121-130*

(MAMS) [6] and [7]. While the classic memories found their operation on multiplications and additions, the MAMS do it in the *min* and *max* operations used in Mathematical Morphology. Several of these models, especially the morphological models are very efficient to recall patterns corrupted either with additive noise or subtractive noise. To overcome this problem, MAMS memories and theirs variants make use of the so-called kernel approach [6]. Kernels for MAMS are however difficult to find [9]. Additionally, if new patterns have to be added to the learning set the kernels need to be recomputed again. In [11], the authors describe a memory model able to handle mixed noise by means of the well-known median operator. Median operation is however time consuming as know. In this paper we show how by decomposing a pattern into its sub-patterns, we can avoid the use of kernels and the median operator. We give numerical and realistic examples where the effectiveness of the proposal is tested.

## 2 Basics About Associative Memories

An associative memory as defined in [13] is an input-output system able to associate an input pattern with an output pattern as follow: $a \rightarrow \boxed{\mathbf{M}} \rightarrow b$, with $a$ and $b$, respectively the input and output patterns vectors. Each input vector forms an association with its corresponding output vector. An association between input pattern $a$ and output pattern $b$ is denoted by $(a,b)$. For $k$ a positive integer, the corresponding association will be $(a^k,b^k)$. Associative memory **M** is represented by a matrix whose *ij*-th component is $m_{ij}$ [2]. **M** is generated from a finite a priori set of known associations, known as *fundamental set of association or simply fundamental set* (FS) [13]. If $k$ is an index, this FS is represented as: $\{(a^k,b^k),k=1,p\}$, with $p$ the cardinality of the set. The patterns integrating a FS are called *fundamental patterns* [13]. The nature of the FS provides an important judgment for associative classification. If for $k=1,p$, it holds that $(a^k=b^k)$, then that memory is auto-associative, otherwise it is hetero-associative [13].

Fundamental patterns could be distorted with noise. A distorted version of a pattern $a$ will be denoted as $\tilde{a}$. If when presenting to an associative memory **M** a fundamental pattern, **M** responds with the correct pattern, we say that **M** presents perfect recall. If for all patterns of a given FS, perfect recall is obtained, **M** is said to present perfect recall.

## 3 Idea of Solution

As already mentioned, the proposal is based on the locality of the noise affecting the pattern, i.e. when the object is decomposed into several parts, some of them will appear more or less affected by noise, some others will not. From these less altered and the unaltered parts is that the whole object is identified. For example in Figure 1(a) we have an image of an object for which a numerical representation (a pattern) has been obtained. In Figure 1(b), we have the same image but distorted with some noise, this of

course affects also its numerical representation. Finally, in Figure 1(c) it is shown the same pattern but decomposed into several parts. By obtaining these set of parts (sub-patterns), as can be appreciated some sub-patterns appear altered, some others no. From these unaltered sub-patterns is that the object can be identified.

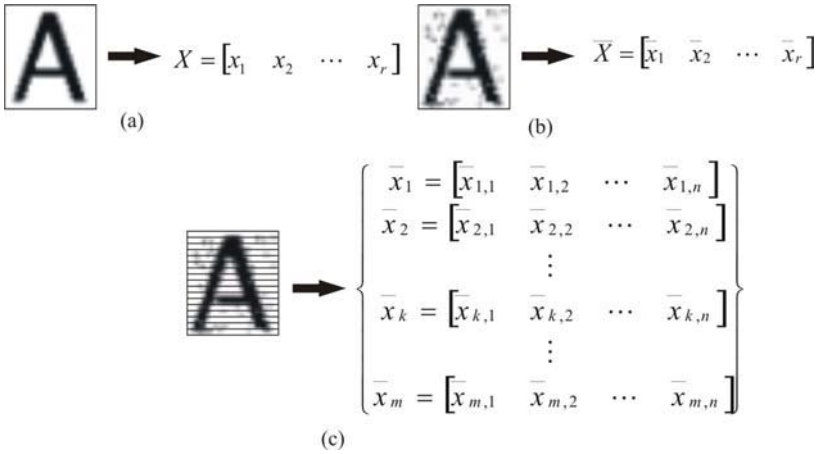



**Figure 1.** (a) An object and its numerical representation. (b) The same object altered by noise and its corresponding numerical representation altered also by the noise. (c) The corresponding pattern decomposed into parts (sub-patterns).

## 4 Basic Definitions

The steps composing the proposed methodology to recognize an object from its parts (sub-patterns) are explained next. For this let us have the following definitions:

**Definition 2.1**. Let $B$ a pattern of an object $O$ obtained somehow (for example as an image-vector by the standard row scanning method or a feature vector). A sub-pattern $b$ of object $O$ is a pattern obtained as $B$ but from a part of $O$.

We have already mentioned that to get the sub-patterns of an object it is necessary to divide this object into parts and obtain their corresponding patterns. We have thus the following definition:

**Definition 2.2.** Let a set of $m$ sub-patterns of an object, represented as row vectors of dimension $n$ denoted by $b_k, k = 1, m$. The matrix $B$ of dimensions $m \times n$ containing all of these patterns as its rows is called *base pattern*.

**Definition 2.3.** The set of all $q$ matrices $B^k$ is called the *base set of matrices* or simply the *base set*, and it is represented as: $\{B^k \mid k = 1, q\}$, with $q$ the number of patterns or objects.

In what follows, for notation purposes $b_k^i$ represents the $k$-th sub-pattern of the $i$-th object, with $k, k = 1, m$ and $i, i = 1, q$.

## 5 The Methodology

The proposal is composed of two main stages: 1) memory training and 2) object recognition. During training the chosen associative memories are built. Also the so-called voting matrix is build. During testing, the objects' sub-patterns are presented to the already trained memories for identification porpoises.

### 5.1 Learning phase

This phase is composed of two main steps as follows:

**Step 1:** For each $k$, take the $b_k^i$ and build associative memory $M^k$. We can select any among the existing different models (see section 6).

**Step 2:** Taking into account that several objects can share a given sub-pattern, we build a matrix $V$ (voting matrix) of dimensions $q \times m$, with $q$ the number of objects and $m$ the number of sub-patterns. Matrix $V$ tells us exactly which sub-pattern is in which object. First row of $V$ is reserved for first object, second row for the second object, and so on. To build $V$, we first fill it of 0's, i.e. $v_{i,j} = 0, i = 1, q; j = 1, m$. We then convert each sub-pattern $b_k^i$ of each base pattern $B^i$ to a decimal equivalent number, and assign this number to component $v_{i,k}$ of $V$. This would mean that sub-pattern $b_k^i$ belongs to base pattern $B^i$. This completes the learning stage.

### 5.2 Recalling phase

We have two cases. First case is related with the recalling of a pattern of the FSP, second case, in the contrary, is focused on the recalling of a pattern of the same FSP but from a distorted version of it.

**Case 1: Recalling a pattern of the FSP.** For each base pattern $B^k$ of the FSP:

**Step 1:** We begin by building a voting vector and filling it by 0's as follows $Z = \begin{pmatrix} z_1 = 0 & z_2 = 0 & \cdots & z_q = 0 \end{pmatrix}$.

**Step 2:** Now, for a given base pattern $B^i$ (it can be anyone of them), for each of its sub-patterns $b_k^i$, we operate it with the corresponding associative memory $M^k, k = 1, m$, convert it to its decimal equivalent, let say $d$. We then look for all the

appearances of $d$ in matrix $V$ at column $k$, and update the corresponding component $z_i$ of vector $Z$ as follows: For $i=1,q$ do $z_i = z_i + 1 \; if \; v_{i,k} = d$. We repeat this process for each sub-pattern of $B^i$.

**Step 3:** We finally get the index of the corresponding pattern as:

$$j = \arg\max_i (z_i), i = 1, q \tag{1}$$

The whole process is repeated for each $B^i$.

**Case 2: Recalling a pattern of the FSP from a distorted version of it.** Given a distorted version $\overline{B}^i$ of one the patterns of the FSP:

**Step 1:** Again, we begin by defining $Z = \begin{pmatrix} z_1 = 0 & z_2 = 0 & \cdots & z_q = 0 \end{pmatrix}$.

**Step 2:** For each sub-pattern $\overline{b}_k^i$ of $\overline{B}^i$, we operate it with the corresponding associative memory $M^k, k = 1, m$, convert it to its decimal equivalent, let say $d$. We then look for all the appearances of $d$ in matrix $V$ at column $k$, and update the corresponding component $z_i$ of vector $Z$ as follows: For $i=1,q$ do $z_i = z_i + 1 \; if \; v_{i,k} = d$. We repeat this process for each base pattern.

**Step 3:** Get the index of the corresponding pattern as:

$$j = \arg\max_i (z_i), i = 1, q \tag{2}$$

### 5.3 Variations

Instead of using the rows of binary patterns to define the base vectors we can use their columns or diagonals and follow the same procedure to learn and recall patterns. The idea is to decompose the pattern into sub-patterns for recalling.

## 6 Numerical Examples

In this section we provide some numerical examples to better understand the functioning of the proposal. In the next section we give some real examples where we test the effectiveness of the proposal.

**Example 6.1.** Let be the following FSP, representing the five vowels of the Latin Alphabet (A, E, I, O and U; 1 is for the information, 0 is for background):

$$B^1=\begin{pmatrix}0&1&0\\1&0&1\\1&0&1\\1&1&1\\1&0&1\end{pmatrix},\ B^2=\begin{pmatrix}1&1&1\\1&0&0\\1&1&0\\1&0&0\\1&1&1\end{pmatrix},\ B^3=\begin{pmatrix}1&1&1\\0&1&0\\0&1&0\\0&1&0\\1&1&1\end{pmatrix},\ B^4=\begin{pmatrix}1&1&1\\1&0&1\\1&0&1\\1&0&1\\1&1&1\end{pmatrix},\ B^5=\begin{pmatrix}1&0&1\\1&0&1\\1&0&1\\1&0&1\\1&1&1\end{pmatrix}$$

**Learning phase:**

**Step 1: Memory construction.** We can use any associative memory. Let us use W associative memory reported in [6] useful to handle with subtractive noise. Just to remember, *W* memories make use of arithmetic subtraction between elements and min ($\wedge$) operator for memory building. For pattern recall they use arithmetic addition and the max ($\vee$) operator. For the details refer to [6]. Because each pattern is composed of five sub-patterns, and each of these sub-patterns is of size 3, we have then the next five memories:

$$W^1=\begin{pmatrix}0&-1&0\\-1&0&-1\\0&-1&0\end{pmatrix},\ W^2=\begin{pmatrix}0&-1&0\\-1&0&-1\\-1&-1&0\end{pmatrix},\ W^3=\begin{pmatrix}0&-1&0\\-1&0&-1\\-1&-1&0\end{pmatrix},\ W^4=\begin{pmatrix}0&-1&0\\-1&0&-1\\-1&-1&0\end{pmatrix},\ W^5=\begin{pmatrix}0&0&0\\-1&0&-1\\0&0&0\end{pmatrix}.$$

For the details of how $W^1$ to $W^5$ were obtained, the interested reader is refereed to [6].

**Step 2: Construction of matrix** $V$. As explained in section 5.1, we proceed with each row of $V$:

For pattern $B^1$ and first sub-pattern $b_1^1=\begin{pmatrix}0&1&0\end{pmatrix}$, $d=2$, thus $v_{11}=2$. For pattern $B^1$ and second base vector $b_2^1=\begin{pmatrix}1&0&1\end{pmatrix}$, $d=5$, thus $v_{12}=5$. If we continue we this procedure for the remaining base patterns of $B^1$ and the sub-patterns of base patterns $B^2$, $B^3$, $B^4$ and $B^5$:

$$V=\begin{pmatrix}2&5&5&7&5\\7&4&7&4&7\\7&2&2&2&7\\7&5&5&5&7\\5&5&5&5&7\end{pmatrix}.$$

This ends the learning stage.

**Recalling phase:**

**Example 6.2.** Recalling a pattern of the FSP. Let us take the first fundamental pattern $B^1$ of example 6.1. Let us proceed:

**Step 1:** As discussed in section 5.2: $Z = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix}$.

**Step 2:** For pattern recall a *W* memory uses arithmetic addition between components and the max ($\vee$) operator. For the details, refer to [6]. Now for each $b_k^1$ of $B^1$, we have:

For $b_1^1 = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$: $W^1 \wedge b_1^1 = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix} \wedge \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} (0+0) \vee (-1+1) \vee (0+0) \\ (-1+0) \vee (0+1) \vee (-1+0) \\ (0+0) \vee (-1+1) \vee (0+0) \end{pmatrix} = \begin{pmatrix} 0 \vee 0 \vee 0 \\ -1 \vee 1 \vee -1 \\ 0 \vee 0 \vee 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$.

$010 = 2_{10}$. We look for the appearances of 2 now into first column of $V$. As can be seen it appears only in the element $v_{11} = 2$ of $V$, so: $Z = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix}$.

For $b_2^1 = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$: $W^2 \wedge b_2^1 = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{pmatrix} \wedge \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} (0+1) \vee (-1+0) \vee (0+1) \\ (-1+1) \vee (0+0) \vee (-1+1) \\ (-1+1) \vee (-1+0) \vee (0+1) \end{pmatrix} = \begin{pmatrix} 1 \vee -1 \vee 1 \\ 0 \vee 0 \vee 0 \\ 0 \vee -1 \vee 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$.

$101 = 5_{10}$. We look for the appearances of 5 now into second column of $V$. As can be seen it appears in the first, fourth and fifth positions of $V$, so: $Z = \begin{pmatrix} 2 & 0 & 0 & 1 & 1 \end{pmatrix}$.

If we continue this way, it can be easily shown that $Z = \begin{pmatrix} 5 & 0 & 0 & 2 & 2 \end{pmatrix}$.

**Step 3:** We finally get the index of the corresponding pattern as: $j = \arg\max_i (5,0,0,2,2) = 1$. Thus the desired pattern is pattern $B^1$, that is the pattern we were looking for.

**Example 5.3.** Recalling a pattern of the FSP given a distorted version of it. Let us now take the following distorting version of fundamental pattern $B^1$:

$$\overline{B}^1 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}.$$

You can observe that in this case sub-patterns $b_1^1$ and $b_2^1$ appear distorted. The other three are not altered. Proceeding as before:

**Step 1:** As discussed in section 5.2: $Z = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix}$.

**Step 2:** Now for each $b_k^1$ of $B^1$, we have:

For $\bar{b}_1^1 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$: $W^1 \wedge \bar{b}_1^1 = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix} \wedge \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} (0+0) \vee (-1+0) \vee (0+0) \\ (-1+0) \vee (0+0) \vee (-1+0) \\ (0+0) \vee (-1+0) \vee (0+0) \end{pmatrix} = \begin{pmatrix} 0 \vee -1 \vee 0 \\ -1 \vee 0 \vee -1 \\ 0 \vee -1 \vee 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.

$000 = 0_{10}$. We look for the appearances of 0 now into first column of $V$. As can be seen, it does not match any element of $V$, so: $Z = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix}$.

For $b_2^1 = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}$: $W^2 \wedge b_2^1 = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{pmatrix} \wedge \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} (0+1) \vee (-1+1) \vee (0+0) \\ (-1+1) \vee (0+1) \vee (-1+0) \\ (-1+1) \vee (-1+1) \vee (0+0) \end{pmatrix} = \begin{pmatrix} 1 \vee 0 \vee 0 \\ 0 \vee 1 \vee -1 \\ 0 \vee 0 \vee 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$.

$110 = 6_{10}$. We look for the appearances of 6 into second column of $V$. As can be seen, again it does not match any element of $V$, so: $Z = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix}$.

If we continue this way, it can be easily shown that $Z = \begin{pmatrix} 3 & 0 & 0 & 1 & 1 \end{pmatrix}$.

**Step 3:** We finally get the index of the corresponding pattern as: $j = \arg\max_i (3,0,0,1,1) = 1$. Again the recalled index coincides with the index of the desired pattern.

## 7 Results with Real Patterns

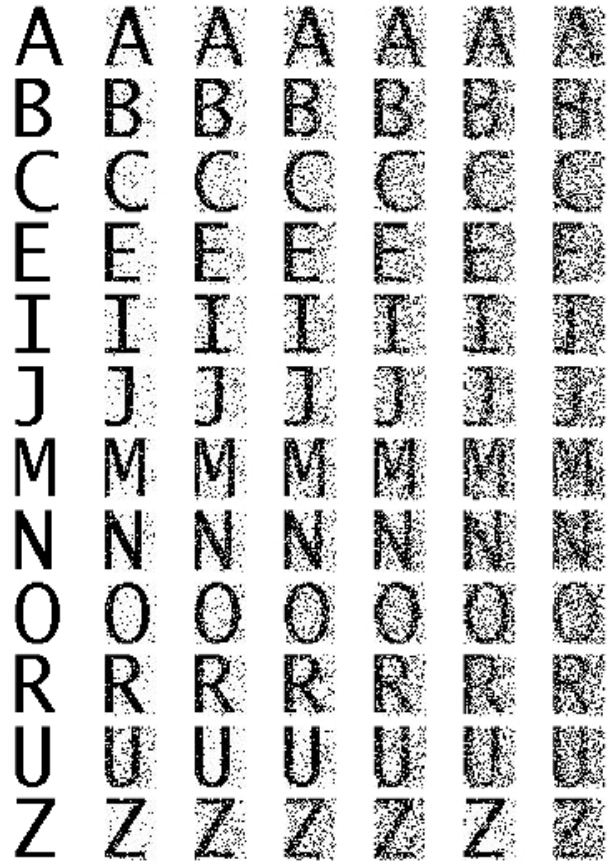
In this section the proposed methodology is tested with more realistic patterns. For this, we make use of the twelve patterns shown in Figure 2. Tests were performed with four associative memories: morphological associative memories **M** and **W** [6], and $\alpha\beta$ associative memories **M** and **W** [8]. Just to remember, **M** memories are good for additive noise and **W** memories are good for subtractive noise. For the details about the operation of both memories, the interested readers is refereed to [6] and [8].

Figure 3 shows the recalling results. As you can appreciate, 100 percent of perfect recall was obtained with **min** (**W**) memories morphological and $\alpha\beta$ from 5 to 15% of noise. From then on, the performance falls little by little. However as can be seen W memories show a better performance than **M** memories.

## 8 Conclusions and Ongoing Research

In this note we have described a simple but effective methodology for the recalling of patterns distorted by mixed noise. Instead of adopting the kernel method used in [6], or the median recently proposed in [11], we prefer to decompose each pattern into a set of sub-patterns. This way we can take advantage of the locality of affecting noise. During memory construction, sub-patterns sets are first used to build a set of memories. Next, during pattern recall a given pattern, possibly distorted by noise is also decomposed into its set of patterns. Each sub-pattern is operated by its corresponding memory. The

filtered result is then used to recover the pattern to which the sub-pattern belongs. Finally, the index of the pattern is recovered by simple voting mechanism.



(a) 0% (b) 5% (c) 10% (d) 15% (e) 20% (f) 25% (g) 30%

**Figure 2.** (a) Patterns used to test the proposal. Their size is 31x37 pixels. For testing these patterns were distorted by mixed noise at percentages of: (b) 5%, (c) 10%, (d) 15%, (e) 20%, (f) 25% and (g) 30% percent. One version of each letter is used.

Nowadays, we are looking for the formal propositions (Lemmas and Theorems and Corollaries) that specify the conditions under which the proposed methodology can be used to perfectly recover a given pattern from a distorted version of it. We are also looking for more real problems where the proposal can find applicability.

**Acknowledgements.** This work was economically supported by CGPI-IPN under grants 20050156 and CONACYT by means of grant 46805. H. Sossa specially thanks COTEPABE-IPN, CONACYT (Dirección de Asuntos Internacionales) and DAAD (Deutscher Akademischer Austauschdienst) for the economical support granted during research stay at Friedrich-Schiller University, Jena, Germany.

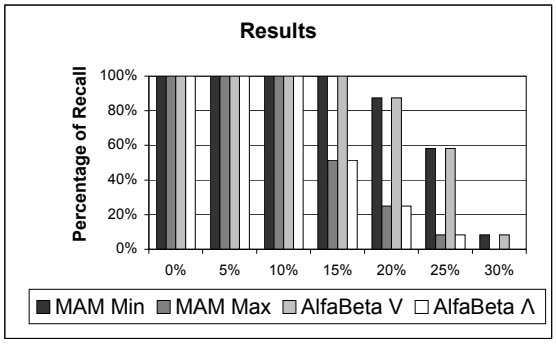



**Figure 3.** Recalling results obtained by applying the proposed methodology to the set of patterns shown in Fig. 2.

## References

1. K. Steinbuch (1961). die Lernmatrix, Kyberneitk C-1,1,26-45.
2. J. Anderson (1972). A simple neural network generating an interactive memory, Mathematical Biosciences C-14, 197-220.
3. T. Kohonen (1972). Correlation matrix memories, IEEE Transactions Computers C-21, 4, 444-445.
4. G. Palm (1982). Neural Assemblies. An alternative approach to artificial intelligence. Studies of the brain function. Springer Verlag.
5. J. Hopfield (1982). Neural Network and Physicals systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, C-79, 2554-2558.
6. G. X. Ritter qt al. (1998). Morphological associative memories, IEEE Transactions on Neural Networks, C-9,281-293.
7. G. X. Ritter et al. (1999). Morphological bi-directional associative memories, Neural Networks, 12:851-867. .
8. C. Yáñez (2002). Associative Memories based on Order Relations and Binary Operators (In Spanish), PhD Thesis, Center for Computing Research-IPN.
9. G. X. Ritter et al. (2003), Reconstruction of patterns from noisy inputs using morphological associative memories. International Journal of Mathematical Imaging and Vision, 19(2), pp. 95-111.
10. H. Sossa et al. (2004). Extended Associative Memories for Recalling Gray Level Patterns. Lecture Notes on Computer Science 3287. Springer Verlag. Pp. 187-194.
11. H. Sossa et al. (2004). New Associative Memories to Recall Real-Valued Patterns. LNCS 3287. Springer Verlag. Pp. 195-202.
12. H. Sossa et al. (2005). Associative gray-level pattern processing using binary decomposition and a-b memories. Neural Processing Letters 22:85-111.
13. M. H. Hassouon (1993). Associated neural memories. Theory and implementation. Oxford University Press.

# Object Classification Based on Associative Memories and Midpoint Operator

Roberto A. Vázquez, Humberto Sossa and Ricardo Barrón
Centro de Investigación en Computación – IPN
Av. Juan de Dios Batíz, esquina con Miguel Otón de Mendizábal
Ciudad de México, 07738, México.
Contact: robertov@sagitario.cic.ipn.mx , hsossa@cic.ipn.mx, rbarron@cic.ipn.mx

**Abstract.** In this paper we describe a way to build an associative memory for object classification. The operation of the new architecture is based on the functioning of the well-know mid-point operator widely used in signal processing. The proposal is an alternative to the one described in [H. Sossa, R. Barrón, R. A. Vázquez. Real-Valued Pattern Classification based on Extended Associative Memory. In Proc. Fifth Mexican Conference on Computer Science (ENC2004), 213-219 (2004). The proposal is tested with image of realistic objects.

## 1 Introduction

One important problem in computer vision is object classification. The solution to this problem would strongly influence the functionality of many systems such us: content-based image retrieval systems, video indexing systems, automatic robot guidance systems, object tracking systems, object manipulation systems, and so on. Many approaches to solve this problem have been proposed in the literature: the well-known statistical approach, the structural approach and the neural approach. The idea of using associative memories to solve the object classification problem is relative new. Refer for example to [1-7].

In this paper we describe an associative model by which we can get the class index of an object given a description of it terms of some its features. We propose a new way to build an associative memory combining well-known set operations of min and max and midpoint operator well-used in signal processing. We show several examples with numerical a real patterns where the effectiveness of the proposal is tested.

Rest of paper is organized as follows. In section 2, the proposal is described in detail. In section 3, a numerical example to better follow the functioning of the proposal is given. In section 4, experimental results with images of realistic objects are provides, while in section 5, conclusions and directions for further research are given.

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 131-140*

## 2 The Proposal

Let $\left(x^{\xi}, i\right)_{\xi=1}^{p}, x^{\xi} \in \Re^{n}, i = 1,\ldots,m$ a set of $p$ fundamental couples (SFC), composed by a pattern and its corresponding class-index. The problem is to build an operator **M**, using this SFC, that allows to classify the patterns into their classes, i.e. $\mathrm{M} \otimes x^{\xi} = i$ for $\xi$=1,…, $p$ and that even in the presence of distortions it classifies them adequately, i.e. $\mathrm{M} \otimes \tilde{x}^{\xi} = i$, where $\tilde{x}^{\xi}$ is an altered version of $x^{\xi}$. A first approach in this direction was presented in [7]. Operator $\otimes$ is chosen such that when operating vector $x^{\xi}$ with matrix **M**, produces as result the corresponding index class of pattern $x^{\xi}$.

Matrix **M** is build in terms of a function $\phi$ as follows:

$$\mathbf{M} = \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_m \end{bmatrix} \tag{1}$$

where each $\phi_i$ represents the $i$-th row of a matrix **M** and this function is a codification of all patterns belonging to the class $i$.

Function $\phi$ can take several forms. In this paper we propose well-known mid-point to build function $\phi$.

### 2.1 Mid-point operator

Arithmetic averaging is widely used in pattern recognition to perform Euclidean distance classification. Arithmetic averaging allows obtaining representative vectors of different patterns to be classified. Representative vectors computed this way are useful only when standard deviation of all patterns belonging to the class $i$, is low.

Mid-point operator usually used in signal filtering could be an alternative to get also representative vectors for each class. We preferred to use mid-point operator because as we will next see it allows better classification results than arithmetic average operator and other known operators.

Mid-point operation works as follows: Given a set of $p$ values: $f_1 \le f_2 \le \cdots \le f_p$:

$$f_{mid} = \frac{f_1 + f_p}{2}. \tag{2}$$

With respect to average operator, mid-point value $f_{mid}$ is always between values $f_1$ and $f_p$, while $f_{ave}$'s position will depend on the distribution of values: $f_1, \cdots f_p$, defined as $f_{ave} = \frac{1}{m}\sum_{i=1}^{m} f_i$.

For the case of vectors, mid-point operator takes the form:

$$\phi_i^j = \frac{\gamma_i^j + \lambda_i^j}{2} \tag{3}$$

where

$$\gamma_i^j = \bigvee_{\xi=1}^{p} \left( x_i^{\xi,j} \right) \tag{4}$$

and

$$\lambda_i^j = \bigwedge_{\xi=1}^{p} \left( x_i^{\xi,j} \right) \tag{5}$$

$i$ stands for the object's class and $j$ goes from 0 to $n$, the size of the pattern. As you can appreciate, the idea is to build a hyper-box enclosing patterns belonging to class *i*, by means of max "$\vee$" and min "$\wedge$" set operators.

**Example 1**. Suppose we want to build matrix $\mathbf{M}_{mid}$ from the following set of associations:

| pattern | class | pattern | class | pattern | class |
|---|---|---|---|---|---|
| (1.0, 1.0, 1.0) | 1 | (4.0, 4.0, 4.0) | 2 | (10.0, 9.0, 10.0) | 3 |
| (1.0, 2.0, 1.0) | 1 | (4.0, 4.0, 5.0) | 2 | (9.0, 9.0, 10.0) | 3 |
| (2.0, 1.0, 1.0) | 1 | (4.0, 5.0, 5.0) | 2 | (10.0, 10.0, 10.0) | 3 |
| (1.0, 1.0, 2.0) | 1 | (5.0, 4.0, 4.0) | 2 | (10.0, 11.0, 11.0) | 3 |
| (2.0, 2.0, 2.0) | 1 | (5.0, 4.0, 5.0) | 2 | (10.0, 9.0, 11.0) | 3 |

According to equations (4) and (5): $\gamma_1 = (2,2,2)$, $\gamma_2 = (5,5,5)$ and $\gamma_3 = (10,11,11)$. Also, $\lambda_1 = (1,1,1)$, $\lambda_2 = (4,4,4)$ and $\lambda_3 = (9,9,10)$. Thus $\phi_1 = (1.5,1.5,1.5)$, $\phi_2 = (4.5,4.5,4.5)$ and $\phi_3 = (9.5,10.0,10.5)$. Finally:

$$\mathbf{M}_{mid} = \begin{bmatrix} 1.5 & 1.5 & 1.5 \\ 4.5 & 4.5 & 4.5 \\ 9.5 & 10.0 & 10.5 \end{bmatrix}.$$

An advantage of mid-point operator over other operators to build matrix **M** is that the distance of representative to farthest class elements is always the same as can be appreciated in Figure 1 (a). For other operators such the well-known arithmetic average operator, representative vector is not always at the center (Figure 1(b)).

Other adavantages of mid-point operator over other operators are the following:

1. It is less expensive to compute a **min** an a **max** than adding up all vectors associatedd to a class as for example with arithmetic average operator.
2. It is less expensive to compute a **min** an a **max** than to order a vector for the case of median operator.
3. It is less expensive to compute a **min** an a **max** than to compute inverse matrices and probabilities as for example with Bayesssian classifier.

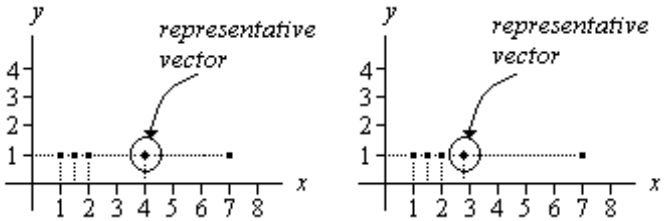



**Figure 1**. (a) Representative pattern obtained by means of mid-point operator is always at the center between farthest class elements. (b) Representative vector obtained by means of arithmetic average operator is not always at central position. This depends on distribution values of class elements.

### 2.2 Pattern classification

Pattern classification is performed as follows. Given a pattern $x^{\xi} \in \Re^n$, not necessarily one of the already used to build matrix $\mathbf{M}_{mid}$, class to which $x$ is assigned is given by

$$i = \mathrm{M} \otimes x^{\xi} = \arg\min_{l} \left[ \bigwedge_{l=1}^{m} \bigvee_{j=1}^{n} \left| m_{lj} - x_j \right| \right] \quad (6)$$

Operators $\vee \equiv \max$ and $\wedge \equiv \min$ execute morphological operations on the difference of the absolute values of the element $m_{lj}$ of $\mathbf{M}_{mid}$ and the components $x_j$ of pattern $x^{\xi}$ to be classified. Thus $\bigvee_{j=1}^{n} \left| m_{lj} - x_j \right|$ is the metric of the max between row $l$ of $\mathbf{M}_{mid}$ and pattern $x^{\xi}$, thus it can be written as $d(x, m_l) \equiv \bigvee_{j=1}^{n} \left| m_{lj} - x_j \right|$, $m_l$ row of $\mathbf{M}_{mid}$.

From the point of view of this metric, pattern classification consists on assigning pattern $x^{\xi}$ to the class which index of row of $\mathbf{M}_{mid}$ is the nearest.

Conditions for correct recall of either a pattern of the FS or from an altered version of one its patterns are given as:

**Theorem 1 [7]**. Let $\mathrm{d_i} = \underset{\text{x class i}}{\vee} \mathrm{d}(\mathbf{x}, \phi_i)$ and $\mathrm{R_i} = \{\mathbf{x} : \mathrm{d}(\mathbf{x}, \phi_i) \le \mathrm{d_i}\}$ hyper-boxes centered at $\phi_i$ and semi-side $d_i, i = 1,...,m$. If $d(\phi_i, \phi_j) > 2\max\{d_i, d_j\}$, then:

i) $R_i \cap R_j = \emptyset, 1 \le \mathrm{i, j} \le \mathrm{m, i} \ne \mathrm{j}$.

ii) $\mathbf{x} \in \mathrm{R_i}$ implies $\mathrm{d}(\mathbf{x}, \phi_i) \le \mathrm{d}(\mathbf{x}, \phi_j)$.

iii) $\mathbf{x} \in \mathrm{R_j}$ implies $\mathrm{d}(\mathbf{x}, \phi_j) \le \mathrm{d}(\mathbf{x}, \phi_i)$.

## 3 Numerical Example

To better understand the idea of the functioning of the proposal, let us study the following numerical example.

### 3.1 Classification of a pattern belonging to the training set

From example 1, let us take pattern $(10.0, 9.0, 11.0)$ that we know it belong to class 3, and let us verify that it is correctly classified. By applying equation 6, we have:

$$l = 1 : \max\left[|1.5 - 10.0|, |1.5 - 9.0|, |1.5 - 11.0|\right] = \max\left[8.5, 7.5, 9.5\right] = 9.5$$
$$l = 2 : \max\left[|4.5 - 10.0|, |4.5 - 9.0|, |4.5 - 11.0|\right] = \max\left[5.5, 4.5, 6.5\right] = 6.5$$
$$l = 3 : \max\left[|9.5 - 10.0|, |10.0 - 9.0|, |10.5 - 11.0|\right] = \max\left[0.5, 1.0, 0.5\right] = 1.0$$

Thus $i = \arg\limits_{l}\left[\bigwedge\limits_{l=1}^{3}(9.5,6.5,1.0)\right] = \arg[1.0] = 3$.

Then the pattern $(10.0,9.0,11.0)$ is assigned to class 3.

### 3.2 Classification of a noisy pattern

From example 1, let us now take distorted version $(9.3,10.5,11.5)$ of pattern $(10.0,9.0,11.0)$ belonging to class 3. Let us verify that in presence of noise, it is assigned to class 3.

$$l = 1 : \max[|1.5 - 9.3|, |1.5 - 10.5|, |1.5 - 11.5|] = \max[7.8,9.0,10.0] = 10.0$$
$$l = 2 : \max[|4.5 - 9.3|, |4.5 - 10.5|, |4.5 - 11.5|] = \max[4.8,6.0,7.0] = 7.0$$
$$l = 3 : \max[|9.5 - 9.3|, |10.0 - 10.5|, |10.5 - 11.5|] = \max[0.2,0.5,1.0] = 1.0$$

Thus $i = \arg\limits_{l}\left[\bigwedge\limits_{l=1}^{3}(10.0,7.0,1.0)\right] = \arg[1.0] = 3$.

Then the pattern $(9.3,10.5,11.5)$ is assigned to class 3.

## 4 Experimental Results

In this section, the proposal is tested with the set of realistic objects shown in Figure 2. Objects were not directly recognized by their images but instead from invariant descriptions of them. With these invariant descriptions matrix $\mathbf{M}_{mid}$ is built. Twenty images of each object in different positions, translations and scaled changes were used to get the invariant descriptions.

(a) (b) (c) (d) (e)

**Figure 2.** The five objects used in the experiments. (a) A bolt. (b) A washer. (c) An eyebolt. (d) A hook. (e) A dovetail.

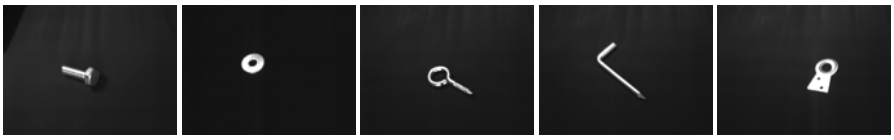
### 4.1 Training phase

To each image of the 20 images of each object a standard thresholder [8] was applied to get its binary version. Small spurious regions from each image were eliminated by means of standard size filter [9]. Next, to each of the 20 images of each object (class) seven well-known Hu geometric moments invariant to translations, rotations and scale changes were computed [10]. After applying methodology described in Section 2, matrix $\mathbf{M}_{mid}$ is:

$$\mathbf{M}_{mid} = \begin{bmatrix} 0.4394 & 0.1598 & 0.0071 & 0.0028 & 1.96E-5 & 0.0011 & -8.47E-6 \\ 0.1900 & 8.72E-5 & 7.47E-6 & 1.28E-14 & 7.23E-14 & -2.93E-10 & -1.6E-14 \\ 0.7092 & 0.2895 & 0.1847 & 0.0730 & 0.0088 & 0.0394 & -0.0015 \\ 1.4309 & 1.6009 & 0.7944 & 0.2097 & 0.0831 & 0.1565 & 0.0118 \\ 0.2475 & 0.0190 & 2.5E-5 & 8.66E-5 & 4.82E-9 & 1.20E-5 & -1.4E-9 \end{bmatrix}$$

### 4.2 Classification

Three sets of images were used to test the efficiency of proposal. A comparison with others proposals was also performed. First set of consisted of 100 images (20 for each of the five objects) different from those used to get matrix $\mathbf{M}_{mid}$. Set number two consisted on other 100 images (20 for each five objects) but projectively deformed. One image of each object is shown in Figure 3, where you can easily appreciate the deformation introduced to the objects. Finally, set number three consisted on other 100 images of five objects (20 for each object) different of those used to get matrix $\mathbf{M}_{mid}$. Figure 4 shows one image of each object. The idea is to verify to which class the object assigned by the classifier.



(a) (b) (c) (d) (e)

**Figure 3**. First image of each object projectively deformed to test performance of proposal. (a) A bolt. (b) A washer. (c) An eyebolt. (d) A hook. (e) A dovetail.

With the first set of images the associative memory built by means of mid-point operator provided 100% of classification. All objects were put in their corresponding class. Thus, performance of proposal was of 100%.

With second set of images, consisting of deformed objects, proposal provided also 100% of efficiency. Again, all objects were correctly sent to their corresponding classes.

**Figure 4**. First image of each object not belonging to the classes of objects to test proposal. (a) Wood bolt. (b) Hook with thread. (c) Open eyebolt. (d) Key. (e) Open S.

| | Bolt | Washer | Eyebolt | Hook | Dovetail |
|---|---|---|---|---|---|
| Wood bolt | - | - | - | 100% | - |
| Hook with thread | - | - | - | 100% | - |
| Open eyebolt | - | - | 45% | 55% | - |
| Key | 25% | - | - | 75% | - |
| Open S | - | - | 100% | - | - |

**Table 1**. Percentage of classification for set number three when proposal is applied.

| | $\mathbf{M}_{prom}$ | $\mathbf{M}_{median}$ | Euclidean | Bayesian | $\mathbf{M}_{mid}$ |
|---|---|---|---|---|---|
| Bolt | 100% | 90% | 100% | 100% | 100% |
| Washer | 100% | 100% | 100% | 100% | 100% |
| Eyebolt | 100% | 100% | 100% | 100% | 100% |
| Hook | 100% | 95% | 100% | 100% | 100% |
| Dovetail | 100% | 75% | 100% | 100% | 100% |

**Table 2**. Comparative classification percentages with respect to other classification schemes when first group of objects is used.

| | $\mathbf{M}_{prom}$ | $\mathbf{M}_{median}$ | Euclidean | Bayesian | $\mathbf{M}_{mid}$ |
|---|---|---|---|---|---|
| Bolt | 100% | 100% | 100% | 100% | 100% |
| Washer | 100% | 100% | 100% | 80% | 100% |
| Eyebolt | 100% | 70% | 90% | 90% | 100% |
| Hook | 100% | 50% | 100% | 100% | 100% |
| Dovetail | 100% | 90% | 100% | 70% | 100% |

**Table 3**. Comparative classification percentages with respect to other classification schemes when second group of objects is used.

For third set of images, Table 1 summarizes the classification results. From this table you can appreciate that in general, the objects were associated to the classes of more similar objects already learned. This experiment was only performed to verify that the proposal sends unlearned objects to their most similar object classes.

Compared to other recently published approaches [7] and classical approaches (Euclidean and Bayesian approach), as can be appreciated in Tables 2 and 3, proposal offers better or competitive classification results, with the advantages already mentioned in Section 2.1.

## 5 Conclusions and Ongoing Research

In this paper, we have proposed a very simple way to build an associative memory based on mid-point operator. It uses **min** and **max** set operations to build memory. Proposal has been tested in different scenarios with images of real objects represented by their moment invariants. Results obtained with proposal are comparable and in some cases better than other proposals as shown in Section 4.

One thing that can be done to probably improve the performance of the proposal is to normalize the values of the invariants so that each feature has the same range of values.

One main drawback of mid-point operator is the presence of outliers in the data. This question will be faced in future works.

Nowadays, we are testing others ways to build function $\phi$, especially when the values of $\phi$ are so close and do not satisfy Theorem 1.

**Acknowledgements.** This work was economically supported by CGPI-IPN under grants 20050156 and CONACYT by means of grant 46805. H. Sossa specially thanks COTEPABE-IPN, CONACYT (Dirección de Asuntos Internacionales) and DAAD (Deutscher Akademischer Austauschdienst) for the economical support granted during research stay at Friedrich-Schiller University, Jena, Germany.

## References

1. K. Steinbuch (1961). Die Lernmatrix. Kibernetik, 1(1):26-45.
2. T. Kohonen (1972). Correlation matrix memories. IEEE Transactions on Computes, 21(4):353-359.
3. J. Hopfield (1982). Neural networks and physical system with emergent collective computational abilities. Proceedings of tha National Academy of Sciences, 79:2554-2558.
4. S. Bandyopadhyay and A. K. Datta (1996). A novel neural hetero-associative memory model for pattern recognition. Pattern Recognition, 29(5):789-795.
5. G. X. Ritter et al. (1998). Morphological associative memories. IEEE Transaction on Neural Networks, 9:281-293.
6. F. Galindo (1998). Matrices Asociativas. Científica, 2(8):17-21.

7. H. Sossa, R. Barrón, R. A. Vázquez (2004). Real-Valued Pattern Classification based on Extended Associative Memory. In proceedings of Fifth Mexican Conference on Computer Science (ENC2004), IEEE Computer Society 213-219.
8. N. Otsu. A threshold selection method from gray-level histograms. IEEE Transactions on SMC, 9(1):62-66 (1979).
9. R. Jain et al. Machine Vision. McGraw-Hill. 1995, pp. 47-48.
10. M. K. Hu. Visual pattern recognition by moment invariants. IRE Transactions on Information Theory, 8:179-187 (1962).

# Associative Processing Applied to Word Reconstruction in the Presence of Letter Scrambling

Humberto Sossa, Ricardo Barrón and Benjamín Torres
Centro de Investigación en Computación-IPN
Av. Juan de Dios Bátiz, esquina con Miguel Othón de Mendizábal
Mexico City, 07738. MEXICO
E-mails: hsossa@cic.ipn.mx, rbarron@cic.ipn.mx, benjamincruz@sagitario.cic.ipn.mx

**Abstract.** In this note we describe how an associative memory can be applied to restore a word to its original "position" given a permutation of its letters. The idea is to first memorize a set of original words with different number of letters. Then the issue is to find the correct word given a permutation of its letters. We provide the formal conditions under which the proposal can be used to perfectly restore the desired word. We also give several examples to show the effectiveness of the proposal.

## 1 Introduction

A very well known word game is the following: Suppose we are given a set $S_w$ of $p$ different words of different cardinality $C(w_i), i = 1,2,\ldots,p$ (by cardinality we mean the number of letters of word $w_i$). From this set, at random, we have a word $w_i$ but with its letters scrambled. After scrambling, some of the letters of the words will remain in their original positions, but some others not. The issue is to find the corresponding original word.

We humans do have a notable capacity to solve problems like this by iteratively exchanging the positions of the scrambled letters and making guesses. An exhaustive rearranging of all possible combinations would allow us to sooner or later find the searched word. By taking into account the orthographical rules of forming words would reduce the searching space. When the number of words, $p$, grows, the complexity of searching also grows.

Associative memories have been used for years to recover patterns from the unaltered or altered patterns keys. See for example [1-9]. In this work we propose to use an associative memory to restore a given word given a permutation of its letters.

## 2 Basics About Associative Memories

As defined by several researchers, an associative memory, denoted as **M** is a device with the capacity to relate input patterns and output patterns: $\mathbf{x} \rightarrow \mathbf{M} \rightarrow \mathbf{y}$, with **x** and **y**, respectively the input and output patterns vectors. Each input vector forms an association with a corresponding output vector.

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 141-150*

An associative memory **M** is represented by a matrix whose *ij*-th component is $m_{ij}$. Matrix **M** is generated from a finite a priori set of known associations, known as the *fundamental set of associations*, or simply the *fundamental set* (FS). If $\xi$ is an index, the fundamental set is represented as: $\{(\mathbf{x}^{\xi}, \mathbf{y}^{\xi}) \mid \xi = 1,2,\ldots,p\}$ with $p$ the cardinality of the set. Patterns that form the fundamental set are called *fundamental patterns*.

If it holds that $\mathbf{x}^{\xi} = \mathbf{y}^{\xi}\ \forall\ \xi \in \{1,2,\ldots p\}$, then **M** is auto-associative, otherwise it is hetero-associative. A distorted version of a pattern **x** to be recalled will be denoted as $\tilde{\mathbf{x}}$. If when presenting a distorted version of $\mathbf{x}^{w}$ with $w \in \{1,2,\ldots,p\}$ to an associative memory **M**, then it happens that the output corresponds exactly to the associated pattern $\mathbf{y}^{w}$, we say that recalling is robust.

## 3 Basics of Median Associative Memories

Median associative memories (MEDMEMs) first proposed in [9], have proven to be very powerful tools to recover patterns from distorted versions of their corresponding keys. Two associative memories are fully described in [9]. Due to space limitations, only hetero-associative memories are described. Auto-associative memories can be obtained simply by doing $\mathbf{x}^{\xi} = \mathbf{y}^{\xi}\ \forall\ \xi \in \{1,2,\ldots p\}$. Let us designate Hetero-Associative Median Memories as HAM-memories.

### 3.1 Memory construction

Two steps are required to build the HAM-memory. Let $\mathbf{x} \in \mathbf{Z}^{n}$ and $\mathbf{y} \in \mathbf{Z}^{m}$ two vectors:

**Step 1:** For each $\xi = 1,2,\cdots,p$, from each couple $(\mathbf{x}^{\xi}, \mathbf{y}^{\xi})$ build matrix: $\mathbf{M}^{\xi}$ as:

$$\mathbf{M}^{\xi} = \begin{pmatrix} \mathrm{A}(y_1, x_1) & \mathrm{A}(y_1, x_2) & \cdots & \mathrm{A}(y_1, x_n) \\ \mathrm{A}(y_2, x_1) & \mathrm{A}(y_2, x_2) & \cdots & \mathrm{A}(y_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{A}(y_m, x_1) & \mathrm{A}(y_m, x_2) & \cdots & \mathrm{A}(y_m, x_n) \end{pmatrix}_{m \times n} \tag{1}$$

**Step 2:** Apply the median operator to the matrices obtained in Step 1 to get matrix $\mathbf{M}$ as follows:

$$\mathbf{M} = \underset{\xi=1}{\overset{p}{\mathbf{med}}} \left[\mathbf{M}^{\xi}\right]. \tag{2}$$

The *ij*-th component **M** is given as follows:

$$m_{ij} = \underset{\xi=1}{\overset{p}{\mathbf{med}}} \mathrm{A}\left(y_i^{\xi}, x_j^{\xi}\right). \tag{3}$$

### 3.2 Pattern recall

We have two cases:

**Case 1: Recalling of a fundamental pattern**. A pattern $\mathbf{x}^w$, with $w \in \{1,2,\cdots,p\}$ is presented to the memory **M** and the following operation is done:

$$\mathbf{M} \lozenge_{\mathrm{B}} \mathbf{x}^w. \tag{4}$$

The result is a column vector of dimension *n*, with *i*-th component given as:

$$\left(\mathbf{M} \lozenge_{\mathrm{B}} x^w\right)_i = \underset{j=1}{\overset{n}{\mathbf{med}}} \mathrm{B}\left(m_{ij}, x_j^w\right). \tag{5}$$

**Case 2: Recalling of a pattern from an altered version of it**. A pattern $\tilde{\mathbf{x}}$ (altered version of a pattern $\mathbf{x}^w$ is presented to the hetero-associative memory **M** and the following operation is done:

$$\mathbf{M} \lozenge_{\mathrm{B}} \tilde{\mathbf{x}}. \tag{6}$$

Again, the result is a column vector of dimension *n*, with *i*-th component given as:

$$\left(\mathbf{M} \lozenge_{\mathrm{B}} \tilde{\mathbf{x}}\right)_i = \underset{j=1}{\overset{n}{\mathbf{med}}} \mathrm{B}\left(m_{ij}, \tilde{x}_j\right). \tag{7}$$

Operators A and B might be chosen among those already proposed in the literature. In this paper we adopt operators A and B used in [5]. Operators A and B are defined as follows:

$$\mathrm{A}(x, y) = x - y \tag{8.a}$$

$$\mathrm{B}(x, y) = x + y \tag{8.b}$$

Sufficient conditions, for perfect recall of a pattern of the FS or from an altered version of them, according to [9] follow:

**Proposition 1 [9]**. Let $\left\{\left(\mathbf{x}^{\alpha}, \mathbf{y}^{\alpha}\right) \mid \alpha = 1,2,\ldots,p\right\}$ with $\mathbf{x}^{\alpha} \in \mathbf{R}^n$, $\mathbf{y}^{\alpha} \in \mathbf{R}^m$ the fundamental set of an HAM-memory **M** and let $\left(\mathbf{x}^{\gamma}, \mathbf{y}^{\gamma}\right)$ an arbitrary fundamental couple with $\gamma \in \{1,\cdots,p\}$. If $\underset{j=1}{\overset{n}{\mathbf{med}}} \varepsilon_{ij} = 0$, $i = 1,\cdots,m$, $\varepsilon_{ij} = m_{ij} - \mathrm{A}\left(y_i^{\gamma}, x_j^{\gamma}\right)$ then $\left(\mathbf{M} \lozenge_{\mathrm{B}} \mathbf{x}^{\gamma}\right)_i = \mathbf{y}_i^{\gamma}, i = 1 \ldots m$.

**Corollary 1 [9].** Let $\left\{\left(\mathbf{x}^{\alpha}, \mathbf{y}^{\alpha}\right) \mid \alpha = 1,2,\ldots,p\right\}$, $\mathbf{x}^{\alpha} \in \mathbf{R}^n$, $\mathbf{y}^{\alpha} \in \mathbf{R}^m$. A HAM-median memory **M** has perfect recall if for all $\alpha = 1,\cdots,p$, $\mathbf{M}^{\alpha} = \mathbf{M}$ where

$\mathbf{M} = \mathbf{y}^{\xi} \lozenge_{\mathrm{A}} \left(\mathbf{x}^{\xi}\right)^{t}$ is the associated partial matrix to the fundamental couple $\left(\mathbf{x}^{\alpha}, \mathbf{y}^{\alpha}\right)$ and $p$ is the number of couples.

**Proposition 2 [9]**. Let $\left\{\left(\mathbf{x}^{\alpha}, \mathbf{y}^{\alpha}\right) | \alpha = 1,2,\ldots,p\right\}$, $\mathbf{x}^{\alpha} \in \mathbf{R}^{n}$, $\mathbf{y}^{\alpha} \in \mathbf{R}^{m}$ a FS with perfect recall. Let $\eta^{\alpha} \in \mathbf{R}^{n}$ a pattern of mixed noise. A HAM-median memory **M** has perfect recall in the presence of mixed noise if this noise is of median zero, this is if $\underset{j=1}{\overset{n}{\mathbf{med}}} \eta_{j}^{\alpha} = 0, \forall \alpha$.

### 3.3 Case of a general fundamental set

In [10] was shown that due to in general a fundamental set (FS) does not satisfy the restricted conditions imposed by Proposition 1 and its Corollary, in [10] it is proposed the following procedure to transform a general FS into an auxiliary FS' satisfying the desired conditions:

**TRAINING PHASE:**

**Step 1.** Transform the FS into an auxiliary fundamental set (FS') satisfying Theorem 1:

1) Make $D = cont$, a vector.
2) Make $\left(\bar{\mathbf{x}}^{1}, \bar{\mathbf{y}}^{1}\right) = \left(\mathbf{x}^{1}, \mathbf{y}^{1}\right)$.
3) For the remaining couples do {
   For $\xi = 2$ to $p$ {
   $\bar{\mathbf{x}}^{\xi} = \bar{\mathbf{x}}^{\xi-1} + D$; $\hat{\mathbf{x}}^{\xi} = \bar{\mathbf{x}}^{\xi} - \mathbf{x}^{\xi}$; $\bar{\mathbf{y}}^{\xi} = \bar{\mathbf{y}}^{\xi-1} + D$; $\hat{\mathbf{y}}^{\xi} = \bar{\mathbf{y}}^{\xi} - \mathbf{y}^{\xi}$ }

**Step 2.** Build matrix **M** in terms of set FS': Apply to FS' steps 1 and 2 of the training procedure described at the beginning of this section.

**RECALLING PHASE:**

We have also two cases, i.e.:

**Case 1:** Recalling of a fundamental pattern of FS:

1) Transform $\mathbf{x}^{\xi}$ to $\bar{\mathbf{x}}^{\xi}$ by applying the following transformation: $\bar{\mathbf{x}}^{\xi} = \mathbf{x}^{\xi} + \hat{\mathbf{x}}^{\xi}$.
2) Apply equations (**4**) and (**5**) to each $\bar{\mathbf{x}}^{\xi}$ of FS' to recall $\bar{\mathbf{y}}^{\xi}$.
3) Recall each $\mathbf{y}^{\xi}$ by applying the following inverse transformation: $\mathbf{y}^{\xi} = \bar{\mathbf{y}}^{\xi} - \hat{\mathbf{y}}^{\xi}$.

**Case 2:** Recalling of a pattern $\mathbf{y}^{\xi}$ from an altered version of its key: $\tilde{\mathbf{x}}^{\xi}$ :

1) Transform $\tilde{\mathbf{x}}^{\xi}$ to $\bar{\mathbf{x}}^{\xi}$ by applying the following transformation: $\bar{\mathbf{x}}^{\xi} = \tilde{\mathbf{x}}^{\xi} + \hat{\mathbf{x}}^{\xi}$.
2) Apply equations (**6**) and (**7**) to $\bar{\mathbf{x}}^{\xi}$ to get $\bar{\mathbf{y}}^{\xi}$, and
3) Anti-transform $\bar{\mathbf{y}}^{\xi}$ as $\mathbf{y}^{\xi} = \bar{\mathbf{y}}^{\xi} - \hat{\mathbf{y}}^{\xi}$ to get $\mathbf{y}^{\xi}$.

In general, the noise added to a pattern does not satisfy the conditions imposed by Proposition 2. The following result (in the transformed domain) state the conditions under which MEDMEMs present perfect recall under general mixed noise [11]:

**Proposition 3 [11]**. Let $\left\{\left(\mathbf{x}^{\alpha}, \mathbf{y}^{\alpha}\right) | \alpha = 1,2,\ldots,p\right\}$, $\mathbf{x}^{\alpha} \in \mathbf{R}^{n}$, $\mathbf{y}^{\alpha} \in \mathbf{R}^{m}$ a fundamental set $\mathbf{x}^{\xi+1} = \mathbf{x}^{\xi} + D$, $\mathbf{y}^{\xi+1} = \mathbf{y}^{\xi} + D$, $\xi = 1,2,\ldots,p$, $D = \left(d,\cdots,d\right)^{T}$, $d = Const$. Without lost of generality suppose that is $p$ odd. Thus the associative memory $\mathbf{M} = \mathbf{y}^{\xi} \lozenge_{\mathrm{A}} \left(\mathbf{x}^{\xi}\right)^{T}$ has perfect recall in the presence of noise if less than $\left(n+1\right)/2-1$ of the elements of any of the input patterns are distorted by mixed noise.

## 4 The Proposal

The proposal to solve the problem described in section 1 is composed of two phases: Construction of the banks of memories and restoration of the word. The steps of each of these two phases are next explained. Also, in which follows, letters of words are represented in decimal ASCII code before further processing. This way letter "A" is represented thus as 65 in decimal ASCII code, letter "B" as 66, and so on.

### 4.1 Phase 1: Construction of the bank of memories

This phase has two steps as follows. Given a set $S_w$ of $p$ different words with different cardinality $C\left(w_i\right), i = 1,2,\ldots,p$:

**Step 1:** Group words according to their cardinality.
**Step 2:** For lowest cardinality $C_{lowest}$ to biggest cardinality $C_{biggest}$:

1. Codify each word as explained.
2. Due to each FS does not satisfy conditions stated by Theorem 1 and Corollary 1, transform corresponding FS to auxiliary fundamental set FS'.
3. Built corresponding memory **M**.

### 4.3 Phase 2: Word restoration

This phase follows four steps. Given a scrambled word:

**Step 1:** Codify word in decimal ASCII code as explained.
**Step 2:** Transform codified version as explained in step 1 of case 2 of recalling phase (Section 3.3).
**Step 3:** Apply equations (**6**) and (**7**) to transformed version.
**Step 4:** Anti-transform recalled pattern to get desired pattern (step 3 of case 2 of recalling phase (Section 3.3)).

## 5 Numerical Example

To better understand the functioning of the proposal, let us suppose that we are given the following two sets of Spanish words grouped by cardinality 4 and 5 as follows:

{Gato, Sebo, Trío} and {Félix, Lanar, Opino}.

Represented in decimal ASCII code, these two sets are as follows:

{(71,97,116,111),(83,101,98,111),(84,114,161,111)}
and
{(70,130,108,105,120),(76,97,110,97,114),(79,112,105,110,111)}.

**Phase 1: Memory construction:**

**Step 1**: Transformation of FS to auxiliary FS: Suppose that $d = 10$:

| First FS. Words Gato, Sebo and Trío | | Second FS. Words: Félix, Lanar and Opino | |
|---|---|---|---|
| Transformed vector | Difference | Transformed vector | Difference |
| (71,97,116,111) | (0,0,0,0) | (70,130,108,105,120) | (0,0,0,0,0) |
| (81,107,126,121) | (-2,6,28,10) | (80,140,118,115,130) | (4,43,8,18,16) |
| (91,117,136,131) | (-7,3,-25,20) | (90,150,128,125,140) | (11,38,23,15,29) |

**Step 2**: Construction of memories:

According to the material exposed in Section 3.1 we have to memories, one for words of cardinality 3 and one for words of cardinality 4:

$$\mathbf{M}^1 = \begin{pmatrix} 0 & -26 & -45 & -40 \\ 26 & 0 & -19 & -14 \\ 45 & 19 & 0 & 5 \\ 40 & 14 & -5 & 0 \end{pmatrix} \text{ and } \mathbf{M}^2 = \begin{pmatrix} 0 & -60 & -38 & -35 & -50 \\ 60 & 0 & 22 & 25 & 10 \\ 38 & -22 & 0 & 3 & -12 \\ 35 & -25 & -3 & 0 & -15 \\ 50 & -10 & 12 & 15 & 0 \end{pmatrix}.$$

**Phase 2:** Word restoration:

**Example 1**: Given altered version Lanra, reconstruct corresponding word (Lanar):

Solution:

**Step 1:** Codification of word: Decimal ASCII code for scrambled version Lanra is: (76,97,110,114,97).

**Step 2**: Transformation of word: By adding difference vector (4,43,8,18,16) to altered version we get transformed vector (step 1 of case 2: Recalling of a pattern from an altered version of its key):

$$(76,97,110,114,97) + (4,43,8,18,16) = (80,140,118,132,116)$$

**Step 3**: Application of corresponding memory transformed version: In this case we apply matrix $\mathbf{M}^2$ and equations (6) and (7) to transformed version. We get:

$$(80,140,118,115,130)$$

**Step 4:** Anti-transformation of recalled pattern to get desired pattern. As explained in section 4.3 this is done by subtracting from recalled pattern corresponding difference vector. In this case vector (4,43,8,18,16). We get:

$$(80,140,118,115,130) - (4,43,8,18,16) = (80,140,118,97,114),$$

which corresponds as you can appreciate to word: Lanar.

## 6 Experimental Results

In this section we show how the proposal described in section 4 can be used to recover a given word from a scrambled version of its letters. For this the set Spanish words shown in Table 1 is used:

| Number of letters per word and words used in the experiments | | |
|---|---|---|
| 5 | 7 | 9 |
| ABRIR | DECIMAL | CASADEROS |
| FÉLIX | FLUVIAL | COLIBRÍES |
| IBIZA | FORMADO | INCOMODEN |
| LANAR | IDIOTEZ | POPULARES |
| OPINO | LINCHAR | VIOLENCIA |
| PANAL | OSTENDE | - |
| RUEDA | SEMANAL | - |
| RUGBY | - | - |
| TRINA | - | - |

**Table 1**. List of words used in the experiments.

### 6.1 Memory construction

Each word is first codified in decimal ASCII as specified. Each set of codified words, beginning by words cardinality 5 and ending with words of cardinality 9, is then transformed to its corresponding auxiliary fundamental set. First codified word of each

set is used to build corresponding associative memory. At the end of the process we end with six matrices: $\mathbf{M}^1$, $\mathbf{M}^2$ and $\mathbf{M}^3$. Matrix $\mathbf{M}^1$ codifies the information of words with 5 letters. Matrix $\mathbf{M}^2$ codifies the information of words with 7 letters, while matrix $\mathbf{M}^3$ codifies the information of words with nine letters.

### 6.2 Recalling of each fundamental set

Each word of each set was transformed and presented to its corresponding memory. Of course, due to Theorem 1 and its Corollary all words were perfectly recalled.

### 6.3 Recalling of a word from a distorted version of it (first experiment)

In this experiment less than 50% of the letters of each word of Table 1 were exchanged. In the case of words of five letters two letters were chosen, in the case of words of seven letters three letters were chosen, and in the case of the words of nine letters four letters were exchanged. One scrambled version of each word was generated. Each scrambled version was processed as described and the results were summarized in Table 2. As can be seen from this table, in all cases the desired word was correctly recalled. This of course is an expected result due to the noise added to the patterns satisfies the conditions for recalling specified by Proposition 3.

### 6.4 Recalling of a word from a distorted version of it (second experiment)

In this experiment more than 50% of the letters of each word of Table 1 were exchanged. In the case of words of four letters two letters were chosen, in the case of words of seven letters six letters were chosen, and in the case of the words of eight letters four letters were exchanged. One scrambled version of each word was generated. Each scrambled version was processed as described and the results were summarized in Table 3. As can be seen from this table, in some cases the desired word was not correctly recalled, in other cases it was. One can ask why of this fact. In the one hand it was simply because the percentage of 50% given by Proposition 3 was surpassed. In the other hand we have to remember that if the noise added to a pattern is median zero, it does not matter how the pattern is altered it should be correctly recalled. This is exactly what is happening in this case. Let us take for example altered version FDOAMOR (decimal ASCII code: 70 68 79 65 77 79 82) of word FORMADO. You can easily verify that the noise added to word DECIMAL (decimal ASCII code: 70 79 82 77 65 68 79) is:

| | | | | | | |
|---|---|---|---|---|---|---|
| 70 | 68 | 79 | 65 | 77 | 79 | 82 |
| 70 | 79 | 82 | 77 | 65 | 68 | 79 |
| 0 | -11 | -3 | -12 | 12 | 11 | 3 |

By arranging the component of last row and by taking the median we have that the median of the noise added to word in ASCII code equals median (-12,-11,-3,0,3,11,12)=0. Thus despite more than 50% of the components of the word are modified correct recalled is obtained. In the remaining cases the word was not correctly recalled because more than 50% of its components were altered and because the noise

added to the word has no median equal to 0. From this we can conclude that Proposition 2 is a stronger than Proposition 3.

| Word | Generated word | Recalled word |
|---|---|---|
| ABRIR | AIRBR | ABRIR |
| FELIX | XELIF | FELIX |
| IBIZA | IAIZB | IBIZA |
| LANAR | LANRA | LANAR |
| OPINO | OIPNO | OPINO |
| PANAL | PALAN | PANAL |
| RUEDA | RAEDU | RUEDA |
| RUGBY | RUGYB | RUGBY |
| TRINA | RTINA | TRINA |

(a)

| Word | Generated word | Recalled word |
|---|---|---|
| DECIMAL | DECLMAI | DECIMAL |
| FLUVIAL | ALUVIFL | FLUVIAL |
| FORMADO | FOOMADR | FORMADO |
| IDIOTEZ | ZDIOTEI | IDIOTEZ |
| LINCHAR | LIACHNR | LINCHAR |
| OSTENDE | ESTONDE | OSTENDE |
| SEMANAL | SAMANEL | SEMANAL |
| - | - | - |
| - | - | - |

(b)

| Word | Generated word | Recalled word |
|---|---|---|
| CASADEROS | CESADAROS | CASADEROS |
| COLIBRÍES | COLIBIRES | COLIBRÍES |
| INCOMODEN | INCOMODEN | INCOMODEN |
| POPULARES | PUPOLARES | POPULARES |
| VIOLENCIA | VIOCENLIA | VIOLENCIA |

(c)

**Table 2.** Recalling results. (a) Words of five letters. (b) For words of 7 letters. (c) For words of 9 letters. In all cases the desired word was correctly recalled.

| Word | Generated word | Recalled word |
|---|---|---|
| ABRIR | ARIRB | ABRIR |
| FELIX | FXLEI | FELIX |
| IBIZA | IZAIB | IBIZA |
| LANAR | LNARA | LANAR |
| OPINO | ONOPI | OPINO |
| PANAL | PLANA | PANAL |
| RUEDA | REAUD | RUEDA |
| RUGBY | RYBUG | RUGBY |
| TRINA | TANIR | TRINA |

(a)

| Word | Generated word | Recalled word |
|---|---|---|
| DECIMAL | LACEDMI | **ABAFJAI** |
| FLUVIAL | UILLVAF | **CIRSFAI** |
| FORMADO | FDOAMOR | FORMADO |
| IDIOTEZ | TEIIZDO | IDIOTEZ |
| LINCHAR | NCLRHIA | LINCHAR |
| OSTENDE | DTNSOEE | **PTUFOEF** |
| SEMANAL | MSEALNA | **QCKALAJ** |
| - | - | - |
| - | - | - |

(b)

| Word | Generated word | Recalled word |
|---|---|---|
| CASADEROS | SACORADSE | CASADEROS |
| COLIBRÍES | RIESBCLIO | COLIBRÍES |
| INCOMODEN | MEDONCNOI | **JODPNPEFO** |
| POPULARES | LAROPUSEP | POPULARES |
| VIOLENCIA | VAICNELOI | VIOLENCIA |

(c)

**Table 3.** Recalling results. (a) Words of five letters. (b) For words of 7 letters. (c) For words of 9 letters.

It is worth to mention than during recall if the value of a recalled letter goes under the value 65 ('A') or above the value 90 ('Z'), this value is to 65 and 90. This way in a recalled word we avoid having symbols different from letters.

## 7 Conclusions and Present Research

In this brief note we have shown how an associative memory can be used to find (recover) a desired word given a scrambled version of it. The scrambled version is first taken to a transformed domain were it can be operated by the corresponding memory. This operation automatically reorders the letters of the scrambled word.

In the general case we do not know from which word a distorted version was obtained. We are actually working through an efficient method that allows to recognize a given from a distorted version of it without having to compare it with all possible words. We are also looking for more real situations were the proposal could find applicability.

**Acknowledgements.** This work was economically supported by CGPI-IPN under grants 20050156 and CONACYT by means of grant 46805. H. Sossa specially thanks COTEPABE-IPN, CONACYT (Dirección de Asuntos Internacionales) and DAAD (Deutscher Akademischer Austauschdienst) for the economical support granted during research stay at Friedrich-Schiller University, Jena, Germany.

## References

1. K. Steinbuch (1961). die Lernmatrix, Kyberneitk C-1,1,26-45.
2. J. Anderson (1972). A simple neural network generating an interactive memory, Mathematical Biosciences C-14, 197-220.
3. T. Kohonen (1972). Correlation matrix memories, IEEE Transactions Computers C-21, 4, 444-445.
4. J. Hopfield (1982). Neural Network and Physicals systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, C-79, 2554-2558.
5. G. X. Ritter qt al. (1998). Morphological associative memories, IEEE Transactions on Neural Networks, C-9,281-293.
6. G. X. Ritter et al. (1999). Morphological bi-directional associative memories, Neural Networks, 12:851-867.
7. H. Sossa et al. (2005). Associative gray-level pattern processing using binary decomposition and a-b memories. To appear in Neural Processing Letters.
8. C. Yáñez (2002). Associative Memories based on Order Relations and Binary Operators (In Spanish), PhD Thesis, Center for Computing Research-IPN.
9. H. Sossa, R. Barrón and R. A. Vázquez (2004). New Associative Memories to Recall Real-Valued Patterns. LNCS 3287. Springer Verlag. Pp. 195-202.
10. H. Sossa and R. Barrón (2004). Transforming Fundamental Set of Patterns to a Canonical Form to Improve Pattern Recall. LNAI 3315. Springer Verlag. Pp. 687-696.
11. H. Sossa and R. Barrón (2005). Median Associatives Memories: New Results. Submmited to CIARP 2005.

# Machine Learning and Neural Networks

# Hybrid strategies and meta-learning: an inquiry into the epistemology of artificial learning

Ciro Castiello and Anna Maria Fanelli

Dipartimento di Informatica, Università degli Studi di Bari,
Via E. Orabona, 4 - 70126 Bari ITALY
{castiello, fanelli}@di.uniba.it

**Abstract.** The problem of developing artificial learning systems cannot be confined in the realm of computer science and researchers in this field are called to face an ambitious question reverberating on several disciplines. A deeper investigation in this sense reveals an intriguing parallelism between conceptual theories of knowledge and mathematical models of intellect. Hybridisation strategies and meta-learning approaches are discussed in conformity with the indications of a comprehensive epistemological inquiry into artificial intelligence.

## 1 Introduction

The progress attained in artificial learning in the last few decades gave rise to the rapid proliferation of several applications, some of them exhibiting commercial software facets. Also, the oncoming development of new branches of research and the continuous broadening of application environments favoured the machine learning appealing to a new generation of young scientists and specialists. However, if we refer to the very ultimate goal of the research in artificial intelligence (AI), addressing the thorough emulation of human capabilities, a still long route remains to be covered. In this direction, it makes sense, at least in an academic perspective, to investigate theoretical models of intellect, in order to deepen our understanding of human and artificial cognitive mechanisms.

This work takes part in the epistemological debate around the validity of artificial learning methods, not only in terms of contingent performance results, but also considering the theoretical assessment of their inherent foundation. Moving from the classical debates about the human intellect, we propose a critical inquiry to highlight the strict connection existing between philosophical constructions and scientific approaches to artificial learning. Putting emphasis on the twofold character of human thought, combining apriority and adaptivity, we analyse the typical conceptual schemes which formalise the common reasoning mechanisms. It should not come as a surprise the resort to the speculative investigations for tackling questions pertaining to the computer science sphere of activity. Actually, it has been observed that the scientific progress would grow faster if the close relationship between mathematical and philosophical concepts is properly understood and esteemed [1]. Bearing in mind this guideline, we draw up a survey of the various concepts of human intellect, steering our research in the realm

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 153-162*

of artificial learning. The resulting epistemological analysis is helpful to show connections among the ideas developed by thinkers and scientists far separated in time and space. Particularly, the problem of learning is tackled in the article, by establishing an extensive definition of learning by induction which is involved with the Hume's predicament concerning the plausibility of generalisations [2]. In the context of AI system development, we underline how the mechanism of hybridisation could produce a synthesis among adaptivity and apriority (similarly to what happened in particular moments of the philosophical debate). Moreover, to escape the riddle of induction, we propose a peculiar epistemological approach, claiming the possibility for inductive processes to justify themselves. In this way, we address the field of *meta-learning* as a promising research direction to vitalize the studies in artificial intelligence. By the end of the paper, we shall be able to briefly illustrate also a particular meta-learning framework developed according to the indications provided by the epistemological inquiry.

## 2 The quest for a theory of knowledge

Since from the ancient times, philosophers tried to find a rationale behind the processes of knowledge acquisition and the mechanisms of learning. A first relevant construction of a theory of knowledge could be traced back to Plato. The Greek philosopher emphasised the inadequacy of human senses, stating that the ability to think is founded on a priori concepts embedded into the human mind, namely the *Ideas* [3]. These abstract concepts are endowed with complete worthiness and possess true and eternal existence. This is the reason why the Platonic philosophical establishment is also referred to as *realism*, indicating the reality of a priori Ideas, opposed to the unreality of experience. Engaging in controversy with the Platonic realism, a different conceptual perspective called *nominalism*, related to the Cynic school of philosophy, stated the impossibility of grasping the universal concepts without the recourse to sensible experience. In this way, the Ideas lose any existential connotation and are regarded only as labels indicating ensembles of objects.

The realism and the nominalism have represented the keystones of the philosophical debates in the subsequent centuries. On the one hand, Platonic realism advocates committing to apriorism for grounding a theory of knowledge, that should always start from a base of eternal immutable universal concepts. On the other hand, nominalism considers the sensible experience as the only source of knowledge, thus resorting to adaptivity for adequately tackling the natural plurality. Plato's principle of apriority provide for an answer about the possibility of knowledge, but it does not suffice to approach another fundamental question concerning intellect: how is *learning* possible? This kind of inadequacy was quite soon identified as a "leak" into the Platonic construction: Aristotle recognised that Plato's formulation cannot generate any form of learning, since Ideas are detached from the world where the universal concepts become incarnate. To solve this problem, the Aristotelian theory is based on the assumption that the communication of Ideas with the physical world is resolved in the meeting be-

tween form and matter [4]. The Aristotelian forms are characterised by an a priori universal reality and represent the formative principle in human learning. However, forms possess also a dynamic nature, being able to originate all the extraordinary variety of the physical world, during their encounter with matter. Aristotle's construction can be seen as the first attempt to approach the debate about the theory of knowledge by combining apriority with adaptivity of mind.

Actually, the divergences between Aristotle and Plato were minimised by the thinkers of the years to come and the lack of clarity in Aristotelian theory contributed to the mediaeval controversy involving the sustainers of nominalism and realism. The theory of knowledge evolved through a debate strongly biased by the role played by theological thinking. The famous "Occam's razor", denying any resort to universal a-priori concepts to attain knowledge of the world, founded the problem of knowledge on direct experience and encouraged the scientific research and the development of the coming philosophy of empiricism.

The epistemological problem assumes the connotation of the "specific problem" of the modern philosophy: rationalism and empiricism can be seen as means to grasp the reality outside the mind. These means share the awareness of mental representations and external reality, but differ in their approaches. Rationalism answers the question of knowledge by highlighting the misleading nature of sensitivity and by proposing a metaphysical construction to bridge the gap between mental representations and external reality. Empiricism underlines the revealing character of sensitivity, trying to learn external reality by questioning our senses (denying any apriority for our mental representations). Again, the opposition between intellectual apriorism and natural adaptivity stands out, with a reprise of the dualism of realism and nominalism. As concerning the problem of learning, the empirical perspective is based on inductive approaches affirming the foundation of the knowledge of the world on simple sensible data. Particularly, Hume stressed the empirical tendencies by examining possibilities and limitations of human cognitive experience. In [2], the Scottish thinker faced the causality problem and, in his argumentations, the idea of a necessary connection cause-effect is ruled out both in aprioristic sense and in relation to any source of experience. In this way, the causality principle is spoiled of necessity, loosing epistemological justification: only habit is responsible of human generalisations. This consideration is pregnant of significance in our inquiry, since it asserts that induction is not a valid form of reasoning and, consequently, a criticism is raised with regard to the scientific method that aims at generalisation.

Similarly to Aristotle, Kant tried to solve the problem of knowledge composing the breach between rationalism and empiricism. In [5] it is shown how, although knowledge cannot transcend experience, nevertheless it is partly characterised by an a priori component that is not inductively inferable from experience. The novel epistemological argumentation asserts that science is based on synthetic judgements a priori that, even if can be evoked by experience, should be founded on a very solid base that induction could never offer to a general law. Again, the keystone of a unifying theory consists in its effort of combining apriority with adaptivity of mind. In the following, we start an analysis of formal

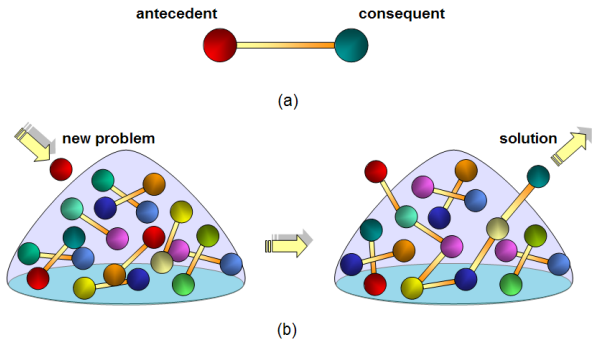



**Fig. 1.** The working mechanism of a rule-based artificial intelligent system.

modelling of intellect to underline the connection existing among mathematical concepts and philosophical positions.

## 3 Mathematical concepts of intellect

When dealing with logical reasoning, a concept can be thought as a rule by which an instance domain is partitioned into a subset of instances satisfying the rule, and another subset whose instances do not satisfy the rule. The process that allows to advance from general concepts to particular rules is referred to as *deduction*. If we intend to design an AI system working on the basis of deductive reasoning, then we should pay attention in providing it with a priori knowledge, namely a database of general concepts and rules, useful for tackling world problems. Deductive approach can be easily brought back to the Platonic realism: the role of a priori principles is emphasised in the knowledge construction process and experience of sensible world is just an afterthought.

When the reasoning process allows us to advance from particular observations to general concepts, then the logical inference performed is referred to as *induction*. An AI system, designed to reason in terms of inductive inferences, needs no a priori content: its action will be driven by the observation of real objects and the consequent generalisation processes. Induction could be related to the nominalistic approach: the role of universal principles is underestimated and general concepts become names, assigned to classes of similar objects.

The deductive inference is characterised by an additive, demonstrative, non-ampliative nature, which is able to preserve truthfulness. On the other hand, inductive learning preserves falsity and does not preserve truthfulness, thus showing a non-additive, non-demonstrative nature. Moreover, induction helps to go beyond deductive attainments, since inductive conclusions entail more information contents than those embedded into the premise concepts.

An analysis of mathematical models of intellect can be focused in the field of artificial intelligence, in particular reviewing rule-based models and connectionist systems. Assuming that a priori knowledge has to be embedded into a

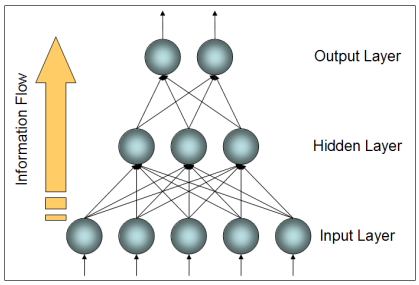



**Fig. 2.** The structure of an ANN composed by two unit layers, plus an input layer.

machine, an AI system can be endowed with a base of logical rules, similar to the high-level cognitive concepts utilised by a human in conscious decision-making processes. The mathematical formalisation of this concept of intellect makes use of name variables, rather then numbers, and logical inferences represent the basis of reasoning and knowledge acquisition. This direction in the theory of intellect has been usually referred to as *rule-based AI* or, with a misnomer, *symbolic AI* [6, 7]. The basic working plan of a rule-based system follows the reasoning process of a human expert to solve a real-world problem. All the possible situations in a particular environment are codified in a number of rules of the typical form: "IF *antecedent* THEN *consequent*". The necessary relationship between *antecedent* and *consequent* formalises in a logical form the knowledge of the expert (as sketched in figure 1(a)). When the system has to face a particular problem, the real occurrence is translated into a logical form, consistent with the knowledge base, and a concatenation of inferences produces the final solution, as depicted in figure 1(b). The question about the possibility of learning is re-proposed in rule-based AI, since the deductive inferences operated by logical systems do not prove to build up a model of intellect with actual knowledge enlargement capabilities. Moreover, combinatorial complexity undermines the foundation of rule-based AI. Actually, systems of logical rules are doomed to perform well in limited domains, since the amount of concepts to be formalised is not prohibitive.

Artificial neural networks (ANN) are computational models that, loosely motivated by biological systems, exhibit some of the properties of the brain [8, 9]. They are composed by a number of simple processors (neurons) working in parallel, without any centralised control. The neurons are arranged into a particular structure (usually organised in layers), where a system of weighted connections guarantees the information flow through the network (see figure 2). Neural networks are commonly regarded as learning machines that work solely on the basis of empirical data. The only means for acquiring knowledge about the world in a connectionist system come from observational instances and there are no a priori designed conceptual patterns that could lead the learning process. The lack of any kind of conceptual cognition and the resort to data for developing knowledge let us review the connectionist approach as an empirical attitude in the context of the theory of intellect. Neural networks have shown their effectiveness

in a number of applications, however they exhibited also a variety of problems that in many cases limit their profitable employment. In particular, the most relevant difficulties are related to the lack of transparency of neural networks (that represents also an obstacle for the a priori knowledge exploitation), and the number of training samples required for learning (that could be prohibitive when dealing with large, complex real-world problems).

## 4 Complexity, hybrid strategies and fuzzy logic

The conducted analysis would suggest that every attempt to develop a comprehensive mathematical model of human intellect could be frustrated by a combinatorial complexity explosion. In fact, methods based on adaptivity are subjected to combinatorial explosion of the training process. On the other hand, approaches related to apriority have to face combinatorial explosion of the knowledge base complexity. A lesson can be derived from those remarks: the matter of combining adaptivity and apriority assumes paramount relevance in artificial intelligence, similarly to what happens in the debates for understanding human intelligence. As already pointed out, Aristotle perceived that the lack of adaptivity would have doomed the Platonic theory of ideas to cut off every kind of learning capacity. Correspondingly, the Kantian construction of a theory of knowledge, based on synthetic judgements a priori, implicitly expressed the urge for a combination of the aprioristic contents of intellect with its adapting capabilities. Also in recent lines of inquiry, related to the field of philosophy of science, the mechanism of hybridisation has been appraised as a more correct attitude for developing consistent research in the AI field [10]. Hybridisation, in fact, appears to be a much more effective practice to produce successful models, in place of the abused appeal to "paradigms" (that disorderly evolve in a quite exaggerated number in AI contexts, with respect to what happens in more consolidated sciences, such as physics).

Nevertheless, the problem of complexity is deeply rooted in some kind of contradictions that can be highlighted once again by referring to the conceptual discussions of the past. Aristotelian logic hardly conciliates with the theory of forms: while the first describes laws governing definitive and eternal truths, the latter emphasises the dynamic and adaptive role of forms in a mutable world. In modern times, Kant operated a "Copernican revolution" to explain the modalities of the knowledge construction process. The novel epistemological assessment, while stating that it is impossible to know the *thing-in-itself* of the world reality, transfers the focus on the cognitive subject and her own peculiar ability of perceiving phenomena. This way of understanding reality could be hardly represented by the logical mechanisms, which seek for the absolute essence of the thing-in-itself. When the mature logical tradition of the early 1900s resolved to eliminate any uncertainty and subjectivity from the knowledge construction process, a definitive impediment disturbed the mathematical dream. Gödel theorems of incompleteness established that the price for the exactness is paid in terms of completeness. A different direction to resolve the Aristotelian contra-

diction, opposing rigid logical schemes to the plasticity of human thought, was undertaken by accepting uncertainty in reasoning process. Fuzzy logic invalidates the cornerstones of formal logic (namely, the law of excluded third and the principle of contradiction) and brings forward a form of approximate reasoning that, while renouncing exactness, fits better the vagueness of real world situations. Into the inherent nature of fuzzy logic, admitting a subjective capability of expressing different degrees of truth, it is possible to trace the echoes of the modern philosophical attitude toward the theory of knowledge, as expressed by Schopenhauer's words. "The world is my representation" [11] could be intended as the *ante litteram* statement of the novel conception of knowledge embedded in the fuzzy way of reasoning.

## 5 Learning through induction

As we have already pointed out, the problem of establishing a proper definition of learning has troubled thinkers and scientists for many times. We assume that learning occurs by increasing the amount of available knowledge, namely by enlarging the base of knowledge determined by a "deductive closure". In order to derive some new pieces of information, the "inductive leap" appears to be a necessary mechanism, therefore we concentrate on induction to properly discuss an epistemological assessment of learning practices.

It is straightforward to relate this kind of definition of learning, connected with induction, with the conceptual disputes dating back to Hume's argumentations about the generalisation plausibility. It should be noted that the intriguing unsafety of induction does not regard only the guarantee of generating correct conclusions: it is also doubtful whether the basic inductive mechanisms possess credibility, in any meaningful sense. The crux of the matter in Hume's argumentation relies in the inability to define a rationale behind inductive activities, since no finite number of observations could be enough reason to suppose anything general. This consideration ultimately prevents the support of any degree of confidence in any prediction. Following this line, only habit (namely, repeated observation of regularities) is responsible for the generalisation practice [2].

The problem of induction represents the starting point also for modern naturalism, suggesting a new attitude to tackle generalisation [12]. Moving from the observation of the defeat of traditional epistemology, that was not able to escape the stalemate of the Hume's predicament, the new claim of naturalism consists in reducing the human knowledge to a natural phenomenon, falling under the activity sphere of science. In this way, epistemological problems become scientific concerns, thus reducing the role of the theory of knowledge: an implicit reshaping of epistemology is applied, admitting the out-of-reach character of traditional investigations. Following the naturalistic view, inductive processes are endowed with the faculty of justifying themselves, and epistemological concerns address a higher conceptual level where basic learning practice leaves room for *meta-learning* investigations. In other words, the attention is shifted from mere justification of induction toward the problem of performing a suitable selection

among inductive hypotheses. Of course, these aspects are intrinsically connected and their examinations cannot be conducted in a separate fashion. Nevertheless, it seems that an interesting approach could be grounded on a modified basic perspective. Instead of reviewing our mind activity as a process triggered off only by the experience of particular regularities in the world, we could think of our inductive mechanisms as a perpetual motion of the mind, which naturally generalises from observations along different lines, and progressively becomes skilled in tracing the correct directions.

In practice, artificial learning, recognised as the empirical science of inductive methods, provides a laboratory to develop and evaluate generalisation strategies. If inductive practices scatter in several directions, then producing successful generalisations is just a matter of defining the proper *bias* which helps to find the way in each circumstance. Meta-learning should be able to provide the necessary knowledge of the world and guidance for determining the proper direction in generalisation processes. The meta-learning activity should rely on the assumption that previously successful strategies of induction are supposed to generate hypotheses which can be generally considered better supported. In other words, predictive success provides one of the most powerful basis to assess inductive conclusions. Following this approach, where inductive practices are evaluated by resorting to induction, meta-learning strategies should be employed in the field of artificial learning, supported by the current directions of epistemological investigations. In the following we are going to take a closer look at the computational methods of artificial learning, briefly reviewing the limitations of common base-learning strategies and the potentialities of meta-learning approaches.

### 5.1 Computational methods and induction: being MINDFUL when learning

The applied research in the field of artificial intelligent systems often deals with empirical evaluations of machine learning algorithms to illustrate the selective superiority of a particular model. This kind of approach, with multiple models evaluated on multiple datasets, is characterised by a "case study" formulation that has been recognised and criticised in literature [13, 14]. The selective superiority demonstrated by a learner in a case study application reflects the inherent nature of the so-called *base-learning* strategies, where data-based models exhibit generalisation capabilities when tackling a particular task. Precisely, base-learning approaches are characterised by the employment of a fixed bias, that is the ensemble of all the assumptions, restrictions and preferences presiding over the learner behaviour. This means a restricted domain of expertise for each learning model, and a reduction in its overall scope of application. The limitations of base-learning strategies can be theoretically established: the no free lunch theorems express the fundamental performance equality of any chosen couple of learners (when averaged on every task), and deny the superiority of specific learning models outside the case study dimension [15].

Obviously, if we want to perform pragmatic investigations of particular domains, base-learning approaches represent a quite satisfactory way of proceeding

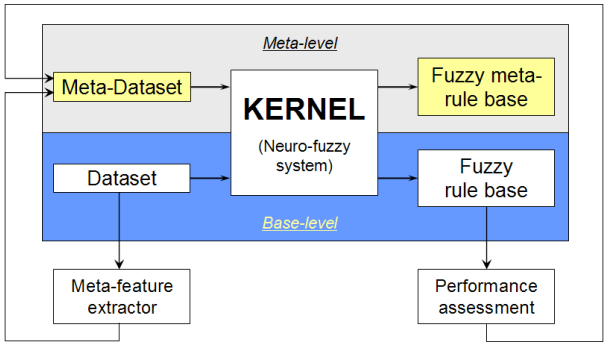



**Fig. 3.** The design of the MINDFUL system; the kernel of the system is represented by a neuro-fuzzy learning scheme.

to obtain adequate results. On the other hand, whenever we are interested in following a line of research with a broader scope, involving more general theoretical issues and some kind of cross-domain applications, the resort to somewhat different methodologies is advisable. By focusing the attention on the role of bias, we characterise the *meta-learning* approach as a dynamical search of a proper bias, that should be able to adapt the learner behaviour to the particular task at hand. The research field of meta-learning represents a novel approach aiming at designing artificial learners with enhanced capabilities, possibly capable of profiting from accumulated past experience [16, 17]. In this way, the formulation of the model evaluation could overcome the case study dimension and the limitations of the base-learning strategies.

The conducted epistemological inquiry, in the way it has been described in this paper, ultimately directed our investigation to design a particular meta-learning framework, namely the MINDFUL (Meta-INDuctive neuro-FUzzy Learning) system, which we are going to synthetically describe. (Obviously, the comprehensive presentation of the MINDFUL system, together with the discussion of its realisation and evaluation do not concern the scope of this article, see [18] for further details.) To compose the schism between aprioristic knowledge representations and adaptive fitting to data observations, our meta-learning methodology is centred on the integration of apriority and adaptivity, conjugating the expressiveness of a rule base with the effectiveness of a neural model. Moreover, this kind of hybridisation takes into account the problem of complexity, and aims at combining the neural network learning capabilities with the representational power of fuzzy logic. In this way, the learning framework is based on the employment of a neuro-fuzzy integration which provides the additional benefit of arranging the available knowledge in a comprehensible and manageable fashion. Actually, the neuro-fuzzy scheme had to be adapted to fulfil the meta-learning requirements. The important point here consists in consenting inductive practices to evaluate past experiences of induction and to project successful generalisations beyond the analysed situations. For this purpose, the MINDFUL system has been

organised in order to employ the same neuro-fuzzy learning scheme both as base- and meta-learner (figure 3 depicts the general scheme of the system). In practice, base-level tasks are tackled following a consolidated approach which exploits neural learning for deriving from data a base of interpretable knowledge, useful for solving each specific problem at hand. At the same time, a meta-learning activity is brought forward, where the same knowledge-based methodology is adopted. In this case, a set of meta-features (describing the properties of tasks) is correlated with the bias configurations adopted during the base-level activity (different learning parameter settings are acknowledged as distinct biases of the system). In this way, the meta-learner provides an explicit meta-knowledge, in terms of fuzzy rules, representing a significant form of high-level information to direct the learning process of the base-learner in novel circumstances.

MINDFUL does not pretend to furnish a definitive solution to the meta-learning questions, neither to stand as an arrival point for our investigation. Nevertheless, it is an attempt toward a systematic study of meta-learning, where hybridisation issues and epistemological grounds are particularly emphasised.

## References

1. Perlovsky, L. I.: Neural Networks and Intellect Using Model-Based Concepts, Oxford Univ. Press (2001)
2. Hume, D.: A treatise on human nature (1740)
3. Plato: Parmenides (IV BC)
4. Aristotle: Metaphysics (IV BC)
5. Kant, I.: Critique of pure reason (1781)
6. Minsky, M. L.: Semantic Information Processing, MIT Press (1968)
7. Newell, A., Simon, H. A.: Human Problem Solving, Prentice-Hall (1972)
8. Bishop, C. M.: Neural Networks for Pattern Recognition, Oxford Univ. Press (1995)
9. Haykin, S.: Neural Networks - A Comprehensive Foundation, Prentice Hall (1999)
10. Cordeschi, R.: Filosofia dell'intelligenza artificiale. In Floridi, L. (editor) Linee di ricerca, SWIF, 525–551 (www.swif.it/biblioteca/lr) (2004)
11. Schopenhauer, A.: The world as will and as representation (1818)
12. Quine, W. V. O.: Epistemology naturalised. In Ontological relativity and other essays (1969)
13. Aha, D. W.: Generalizing from case studies: a case study. In Proc. 9th Int. Conf. on Machine Learning (1992)
14. Brodley, C.: Addressing the selective superiority problem: automatic algorithm/model class selection. In Proc. 10th Int. Conf. on Machine Learning (1993)
15. Wolpert, D. H., Macready, W. G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1) (1997) 67–82
16. Thrun, S., Pratt, L., (eds.): Learning to Learn. Kluwer Academic Publisher (1998)
17. Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. Artificial Intelligence Review **18** (2002) 77–95
18. Castiello, C.: Meta-Learning: a Concern for Epistemology and Computational Intelligence. PhD Thesis. University of Bari - Italy (2004)

# Comparison of Neuro-Fuzzy Systems with a Defuzzification-Based Algorithm for Learning Fuzzy Rules

Jean J. Saade and Adel Fakih

ECE Department, FEA, American University of Beirut,
P.O.Box: 11-0236, Riad El Solh 1107 2020, Beirut, Lebanon
E-mail: jsaade@aub.edu.lb

**Abstract**. This study deals in its first stage with some neuro-fuzzy algorithms used to learn fuzzy inference systems. Two categories of neuro-fuzzy learning approaches are described and compared. The first contains the conventional learning approaches, which were developed by Ichihashi, Nomura, Wang and Mendel. The second consists of another method developed by Shi and Mizumoto. The comparison is based on practical properties related to structure and parameters learning. Then, the drawn conclusions and the mentioned properties are used to provide a comparison between the considered neuro-fuzzy methods and a developed defuzzification-based learning algorithm for fuzzy systems. The advantages of this algorithm over the neuro-fuzzy ones are clearly emphasized.

## 1 Introduction

Due to the importance of fuzzy inference systems in the linguistic representation of human knowledge and expertise, the design of these systems has been given a great deal of attention in the literature. Design methods using data-driven neuro-fuzzy learning approaches, where a neural network learning procedure is used to identify Takagi-Sugeno-Kang (TSK) fuzzy model parameters, have been devised [1-7].

This study addresses first two categories of neuro-fuzzy learning approaches: The conventional ones and another new approach [1-6]. An overview of these learning methods is given and then they are compared using structure and learning-related properties. Based on the comparison results and the mentioned properties, a defuzzification-based learning algorithm for fuzzy systems [8-10] is brought into picture and compared with the considered neuro-fuzzy methods. The advantages of this algorithm as they relate to practically important properties; such as the simplicity of setting the initial fuzzy system, the avoidance of non-firing states, linguistic interpretability, etc., are emphasized.

## 2 TSK Fuzzy Inference Models

In a TSK fuzzy system of zero order [7], the antecedent part of each rule is composed of linguistic variables and the consequent is a crisp value. Hence, in a system with $p$ inputs, $x_j$, j=1,2, …, $p$, and one output, y, the rth rule, $1 \le r \le k$, is expressed as follows:

$$R_r : IF\ x_1\ is\ A_{1r}\ and\ x_2\ is\ A_{2r}\ and\ \ldots and\ x_p\ is\ A_{pr}, THEN\ y\ is\ y_r. \qquad (1)$$

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 163-172*

In (1), $A_{jr}$ are the rth rule fuzzy sets assigned respectively over the input variables $x_j$ and $y_r$ is the rth rule crisp consequent. Based on the rules structure, the number of membership functions (MF's) on each input variable is equal to the number of rules and each MF on an input variable participates in only one rule.

The output value corresponding to input vector $\underline{x}_i = (x_{1i}, x_{2i}, \ldots, x_{pi})$ is computed using a weighted average formula as follows:

$$y_i = \sum_{r=1}^{k} h_{ri} y_r \Big/ \sum_{r=1}^{k} h_{ri} . \tag{2}$$

The use of product for "and," which is applied in neuro-fuzzy methods, gives the firing strength of rule r expressed as:

$$h_{ri} = \prod_{j=1}^{p} A_{jr}(x_{ji}) . \tag{3}$$

## 3 Neuro-Fuzzy Learning Methods

Two main neuro-fuzzy learning approaches are of interest here: the conventional one developed in [1-4] and the new approach [6].

### 3.1 Conventional Neuro-Fuzzy Methods

Referring back to Section 2, we note here that Eq. (2) was used by Wang and Mendel [4] and Nomura [3]. Ichihashi [1,2] used a simplified version of (2) to get the system output:

$$y_i = \sum_{r=1}^{k} h_{ri} y_r . \tag{4}$$

Ichihashi and Wang-Mendel used Gaussian MF's while Nomura used triangular ones.

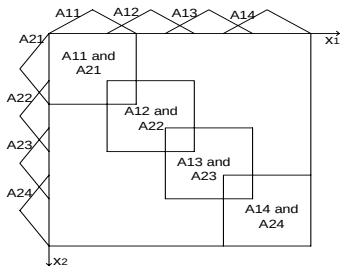



**Fig. 1.** Conventional neuro-fuzzy system with non-firing states

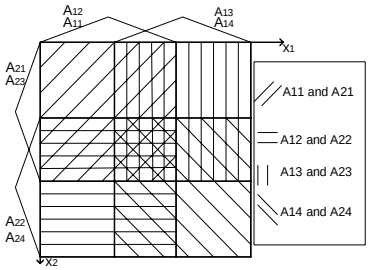



**Fig. 2.** Conventional neuro-fuzzy system with no non-firing states

To avoid initial non-firing states, the firing strength of at least one rule (See (3)) and for any input $\underline{x}_i$ must differ from zero. Hence, any region in the input space must be covered by all the MF's of at least one rule. This requires a special setting of the initial MF's. For a system with two inputs, say, there has to be as many MF's having the same shape on each input as there are distinct ones. Fig. 1 illustrates the case of a 4-rule and 2-input system where non-firing states exist if $\underline{x}_i$ is anywhere in the regions outside those assigned for the rules. In Fig. 2, however, non-firing states are avoided since the MF's are assigned as required. Undoubtedly, the process of initial MF's and rules assignments gets more difficult when the number of these MF's and rules increases and when the system has more than two inputs (See Section 3.3.4).

When n training input-output data ($\underline{x}_i$, $y_{id}$), where i = 1,2,…n, are given, then they are used in an error back-propagation learning to modify the parameters of an initial fuzzy inference system, whose form is given in (1), and minimize the data approximation error. The following error function is usually adopted:

$$E_i = (y_{id} - y_i)^2 / 2. \quad (5)$$

The center and width of triangular MF's, the mean and variance of Gaussian ones and the crisp consequents are updated by the gradient-descent method:

$$a(t+1) = a(t) - \alpha[\partial E_i(t) / \partial a] = a(t) + \alpha[y_{id} - y_i(t)]\, \partial y_i(t) / \partial a \ , \quad (6)$$

where α is a learning rate, t denotes the current iteration and $a$ is the parameter of concern .

Once a data pair ($\underline{x}_i$ , $y_{id}$) is presented to the system and the system parameters are updated based on (6), then the system output for the same input $\underline{x}_i$ changes at each update and also the error $E_i$. The tuning of the system parameters for the input data $\underline{x}_i$ stops when the step size, $d_i = | E_i(t+1) - E_i(t)|$, between two consecutive iterations drops below a given threshold. Then another data pair is presented to the system and the procedure is repeated. When all the data are presented to the system (learning epoch), the total error E is calculated as follows:

$$E = 2\sum_{i=1}^{n} E_i / n = \sum_{i=1}^{n} (y_{id} - y_i)^2 / n \, . \quad (7)$$

If this error is smaller than some desired error, $E_d$, the learning stops. If not a new learning epoch begins. However, the performance of repeated epochs would not necessarily lead to $E \leq E_d$. Hence, the number of epochs is also be considered as a stopping criterion.

In the above-described type of learning (pattern mode), the tuning of the system by a given data ($\underline{x}_i$ , $y_{id}$) affects the tuning of the system by all the subsequent data points. This effect is absent in the batch learning mode since the parameters are updated only after the whole data set is used. Actually, for a point ($\underline{x}_i$,$y_{id}$), the adjustment $\Delta_i a = a(t+1) - a(t)$ of a given parameter $a$ is still computed as in (6). But, this adjustment is stored. When all the data pairs have been used, the total adjustment is computed as $\Delta a = 2\sum_{i=1}^{n} \Delta_i a / n$. In fact, batch learning is equivalent to the use of (6) with $E_i$ replaced by $E$ .

### 3.2 New Neuro-Fuzzy Approach

The major difference between this approach (Shi and Mizumuto [6]) and the conventional ones is that all the combinations of the MF's assigned over the input variables are used to form the antecedents of the rules. This difference along with pattern learning entails modifications in the properties of the algorithm. An overview of the new approach is provided for a system with two inputs.

Let $A_{1s}$, s = 1,2, …, $l_1$ and $A_{2q}$, q = 1,2, …, $l_2$, be the MF's on input variables $x_1$ and $x_2$ respectively. Then, $k = l_1 \times l_2$ fuzzy rules are constructed in the form:

$$\text{Rule } (s-1)l_2 + q\text{: If } x_1 \text{ is } A_{1s} \text{ and } x_2 \text{ is } A_{2q}, \text{ THEN y is } y_{(s-1)l2+q}. \quad (8)$$

With $h_{[(s-1)l2+q]i}=A_{1s}(x_{1i})\times A_{2q}(x_{2i})$ denoting the firing strength of the rule in (8), then the output is calculated as follows:

$$y_i = \sum_{s=1}^{l_1}\sum_{q=1}^{l_2} h_{[(s-1)l_2+q]i} y_{(s-1)l_2+q} \Big/ \sum_{s=1}^{l_1}\sum_{q=1}^{l_2} h_{[(s-1)l_2+q]i} \tag{9}$$

### 3.3 Properties-Based Comparison

The conventional and new neuro-fuzzy learning approaches are compared here based on properties related to their structure and the applied learning procedure.

#### *3.3.1 Type of Membership Functions*

Both conventional and new neuro-fuzzy methods require that the MF's be differentiable with respect to their parameters. This is due to the gradient-descent method (6). Also, any change in the form of the used MF's requires that the parameters updating formulas be rederived.

#### *3.3.2 Type of Logic Operations and Error Function*

The use of Eqs. (2)-(4) and (9) in neuro-fuzzy means that the fuzzy AND, OR and THEN are respectively represented by product, sum and product. This is essential for the gradient-descent formula (6), and the involved derivative. Also, the adopted error function influences the parameters updating formulas.

#### *3.3.3 Type of Learning*

The considered neuro-fuzzy algorithms, use pattern learning (Section 3.1). Referring to Fig. 2 and (2)-(6) it can be seen that more than one training example affect the same system parameters. Hence, by adjusting these parameters based on a data point and then going to the next, should lead to a compromise between the parameters and their affecting points that is not as good as the one obtained using the batch mode of learning (Section 3.1). In fact, this aspect becomes more serious when more data points affect the same system parameters, as in the new neuro-fuzzy approach (Section 3.2) where the MF's are less localized. Hence, the batch mode of learning should be more suitable for the type of fuzzy inference structure used in the new neuro-fuzzy method.

#### *3.3.4 Setting of Initial MF's and Rules*

The major concern in the conventional methods is the avoidance of initial non-firing [5]. Hence, on each system input, there must be a number, $d_{mf}$, of distinct MF's with each repeated $r_{mf}$ times to give rules antecedents covering the whole input space. As explained in Section 3.1, this is not simple especially when the number of rules, $k$, and input space dimension, $p$, increase. To make things easier, two formulas are set. With $d_{mf}\times r_{mf}=k$ and $r_{mf}=d_{mf}\times(p-1)$, then

$$d_{mf} = \sqrt{k/(p-1)} \ \text{and}\ r_{mf} = \sqrt{k(p-1)}\ . \tag{10}$$

Of course, $p$ is application-dependent. Hence, $k$ needs to be chosen to give integer $d_{mf}$ and $r_{mf}$. In the new neuro-fuzzy approach, the assignment of initial MF's and rules is

simple. As long as the adjacent MF's on each input overlap (Fig. 3), then the coverage of the entire input space is guaranteed and initial non-firing is avoided (Section 3.2).

### *3.3.5 Simplicity of Learning Formulas*

The learning formulas in the conventional and new methods are determined using (6). In the conventional methods, each MF is used once in the rules. This makes the application of (6) with $y_i$ as in (2) or (4) and, thus, the learning formulas simple and easily extended to systems with a high number of inputs as compared to the formulas in the new approach, where each MF on a specific input is used with all the combinations of MF's on the remaining inputs (See (9) and also [6]).

### *3.3.6 Number of Tuning Parameters*

For a given number of rules, $k \geq 2$ , and a number of input variables, $p \geq 2$ , the number of tuning parameters in the conventional approach , $(2p+1)k$, is greater than that in the new approach, which is given by $2(l_1+l_2+...+l_p)+k$, where $l_j$ is the number of membership functions on input $x_j$. This can be verified as follows: Since $l_1 \leq k$, $l_2 \leq k, ..., l_p \leq k$, with equalities that cannot be satisfied simultaneously except for: (a) k = 1 for any p, (b) p=1 for any k, then for $k \geq 2$ and $p \geq 2$, $l_1+l_2+...+l_p < pk$. In cases (a) and (b), which rarely occur in practice, the conventional and new neuro-fuzzy approaches have the same number of parameters.

### *3.3.7 Fitting toTraining Data*

A MF in the new neuro-fuzzy method covers a larger area in the input space as compared to the conventional approach (Compare Figs. 2 and 3). Hence, for the same set of data points, the 2 parameters of a MF in the new approach need to be adjusted to accommodate a larger number of data. Consequently, the fitting to training data in the new approach is less precise than that in the conventional one for the same number of rules. Data fitting results are provided in Section 3.3.8.

### *3.3.8 Linguistic Interpretability*

Based on the studies in [11,12], The linguistic interpretability problem in neuro-fuzzy learning relates to the highly overlapping and complex MF's obtained after training. This prevents the simple assignment of linguistic labels to these MF's and leads to the generation of rules lacking a clear linguistic meaning. The issue of tradeoff between precision and interpretability has also been noted in the mentioned studies.

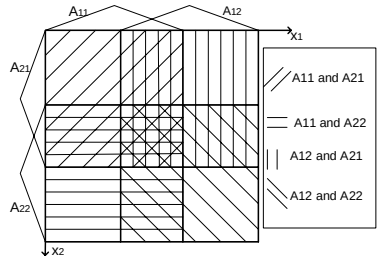



**Fig. 3.** Initial MF's and rules for the new neuro-fuzzy approach

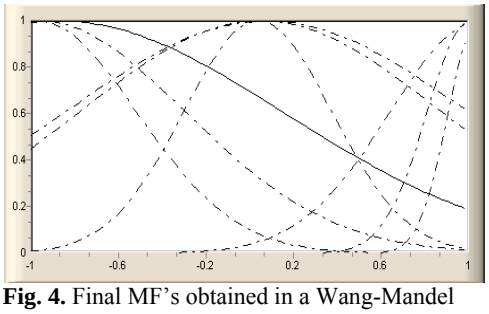



**Fig. 4.** Final MF's obtained in a Wang-Mandel system

The main reason behind the MF's complexity obtained in the neuro-fuzzy methods relates to the unconstrained learning of the MF's parameters. As can be seen in Figs. 2 and 3, the parameters of the MF's assigned on an input variable are changed based on common data points; i.e., located in overlapping regions of the input space, and also on separate data points. Hence, these MF's tend to pass each other, exchange positions, etc., as shown in Figs. 4 and 5. This would hinder the linguistic interpretability of the final fuzzy system. The use of crisp rules consequents does also contribute to the deterioration of the linguistic interpretability aspect.

Fig. 4 shows the final MF's obtained over input $x_1$ after training a 9-rule neuro-fuzzy, Wang-Mendel system using 81 data points retrieved from the non-linear function used in [6] and given below. The least data approximation error E=0.000612 was obtained after performing 100 epochs. The MF's interpretability did not improve for a smaller number of epochs.

$$y = (2x_1 + 4x_2 + 0.1)^2 / 37.21 \ , \ -1 \le x_1, x_2 \le 1 \ . \tag{11}$$

Fig. 5 shows the final MF's over input variable $x_1$ for a 9-rule fuzzy system trained by the new neuro-fuzzy approach and the same data used in Wang-Mendel's method. The least data approximation error E=0.00194 was obtained after performing 16 epochs. After epoch 16, the error value got bigger and no improvement was obtained in the MF's interpretability.

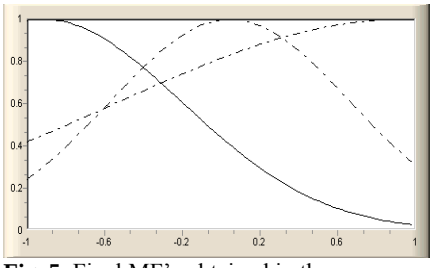



**Fig. 5.** Final MF's obtained in the new neuro-fuzzy method

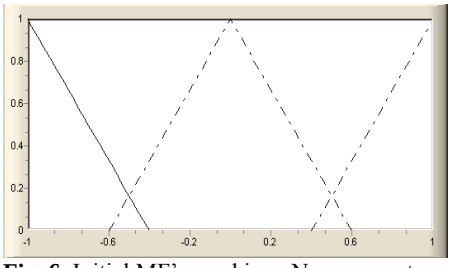



**Fig. 6.** Initial MF's used in a Nomura system

### *3.3.9 Firing State Problem*

In the considered neuro-fuzzy methods, the learning process changes the parameters of the MF's and even duplicated ones (conventional) become distinct during or after learning (See [5]). Hence, even if the initial MF's are as in Figs. 2 or 3, they may turn out to be similar to those in Fig. 1 or having no overlap between adjacent ones due to the unconstrained learning. This causes non-firing states. Hence, the learning may not complete the specified number of epochs. Fig. 6 shows the initial MF's used on $x_1$ and $x_2$ in a 9-rule Nomura system trained using the 81 data points noted in Section 3.3.8. The learning stopped after the second epoch.

## 4 Defuzzification-Based Learning

A new defuzzification-based algorithm for learning fuzzy rules [8-10] is first summarized here, and then, compared with the considered neuro-fuzzy ones. The comparison is based on the properties addressed in Section 3.3.

### 4.1 Defuzzification-Based Algorithm

Consider a two-input, one-output fuzzy inference system. Let $A_{1s}$, s = 1,2…$l_1$, and $A_{2q}$, q = 1,2,…,$l_2$ be overlapping MF's assigned on input variables $x_1$ and $x_2$ respectively and in a manner that the specified ranges of these variables are covered. Then, $k=l_1 \times l_2$ fuzzy rules are constructed as in (8) but with $y_{(s-1)l2+q}$ replaced by overlapping MF's assigned on the output variable y and denoted as $C_{(s-1)l2+q}$ for $1 \leq [(s-1)l_2+q] \leq l_1 \times l_2$. These MF's do not need to be all distinct but they have to cover the entire specified range of the output variable.

The fuzzy output, corresponding to a crisp input pair $\underline{x}_i=(x_{1i}, x_{2i})$, is obtained using the CRI [13]:

$$C_{0i}(y) = \max_{1 \leq [(s-1)l_2+q] \leq l_1 \times l_2} [A_{1s}(x_{1i}) \wedge A_{2q}(x_{2i}) \wedge C_{(s-1)l_2+q}(y)] \quad . \tag{12}$$

The fuzzy OR, AND and THEN are represented here by maximum, minimum and minimum respectively. Other operations can be used as well and (12) can be generalized easily to systems with higher dimensional input spaces. Now, defuzzification applies to the normalized version of $C_{0i}(y)$ , denoted $C_{0in}(y)$, as [8-10]:

$$F_{\delta}[C_{0in}(y)] = \int_0^1 [\delta c_1(\alpha) + (1-\delta) c_2(\alpha)] d\alpha \quad . \tag{13}$$

$[c_1(\alpha),c_2(\alpha)]$ is the α-level set of $C_{0in}(y)$ and δ is a parameter whose values are in [0,1]. Eq.(13) is used to train initial fuzzy systems based on input-output data. All initial rules consequents are required to be equal to the left-most output fuzzy set, which is to be formed by a flat and a decreasing part or a decreasing part only.

Given the training input-output data ($\underline{x}_i$, $y_{id}$), with $\underline{x}_i = (x_{1i}, x_{2i}, \ldots, x_{pi})$, and $\underline{x}_i$, $y_{id}$ being within the specified input and output ranges, the learning starts with an initial fuzzy system as specified above. The algorithm computes the fuzzy outputs for all $\underline{x}_i$'s using (12) and then defuzzifies their normalized versions using (13) with δ=1. Here, due to the above-noted initial rules consequents, all the defuzzified values will be equal to the smallest value of the output range. Hence, $F_1[C_{0in}(y)] \leq y_{id}$ for all i=1, 2, …, n. For these defuzzified values, the total error E is computed using some error function, which could be as in (7) or any other function, and compared with a desired error $E_d$. If $E \leq E_d$ , then the learning stops. Otherwise, δ is decreased from 1 to 0 passing by discrete values. For each $\delta$, the error is computed and compared with $E_d$. The decrease in δ causes an increase in the defuzzified values. They are then made closer to the desired outputs. Whether the change in δ satisfies the error goal, then the learning stops. Otherwise, the algorithm starts another learning round (or epoch) from δ = 1 but with new rules.

These new rules are obtained by raising each rule consequent by one fuzzy set. If this leads to a violation of $F_1[C_{0in}(y)] \leq y_{id}$, it can be reestablished by repeatedly lowering the consequents of the rules triggering one fuzzy output with defuzzified value greater than its desired counterpart. Once the inequality is reinstated, then the decrease in $\delta$ is repeated and the error is computed and compared with $E_d$ . This process is repeated until either the error goal is satisfied or no more raise in the rules consequents is

possible or when the raise and lowering of the rules consequents result in a previously obtained system. When the learning ends, the algorithm delivers the final fuzzy system, the resulting error and the final δ value. A complete description and justification of the learning steps in this algorithm was offered in [9].

### 4.2 Properties-Based Comparison with the Neuro-Fuzzy Algorithms

#### *4.2.1 Type of Membership Functions*

Unlike the considered neuro-fuzzy learning methods, the defuzzification-based algorithm can accommodate any type of MF's. This is because the learning is based on the use of (12) and (13) with no derivatives involved. Also, changing the form of the MF's does not require new formulas for learning.

#### *4.2.2 Type of Logic Operations and Error Functions*

Again, since no derivatives are used, then there is no restriction on the use of operations for AND, OR and THEN as in the considered neuro-fuzzy approaches. Furthermore, since the error function is not differentiated, then any error function; such as, the mean-square error, (7), root mean-square error, mean absolute error, etc., can be used.

#### *4.2.3 Type of Learning*

The objective of the learning process applied in the deffuzification-based algorithm is to reduce the total error resulting from the whole data set rather than the point-wise error. So, the type of learning applied here is compatible with batch mode. This is preferable due to the existence of only one parameter and a fixed number of output fuzzy sets from which the choice is made to form a good compromise for all the data points (See Sections 3.1 and 3.3.8).

#### *4.2.4 Setting of Initial MF's and Rules*

As explained in Section 4.1, the setting of the rules antecedents is easy and is done in the same way as in the new neuro-fuzzy approach. Hence, with overlapping MF's over each input, initial non-firing is avoided (See Eq. (12) and Section 3.3.4). Further, the initial rules consequents are equal to the left-most of the fuzzy sets assigned over the output. This guarantees that for $\delta=1$ the defuzzified output for any crisp input be equal to the lowest value of the output range. Requiring also that the right-most of the fuzzy sets assigned over the output be formed by a flat and an increasing part or an increasing part only guarantees that no defuzzified output for any input and any $\delta\in[0,1]$ exceeds the highest value of the output range. These can be checked easily by referring to (12), (13). In the considered neuro-fuzzy approaches, however, it is not specified how the initial crisp consequents are assigned. Also, we do not have bounds on the system outputs nor specified values for the range of the output variable.

#### *4.2.5 Simplicity of Learning Formulas*

The defuzzification formula (13) is the one used for learning and it applies to the output fuzzy set after it is determined using (12). Hence, the learning formula remains simple even if the dimensionality of the system input or the number of rules increases.

#### *4.2.6 Number of Tuning Parameters*

In the defuzzification-based algorithm, there is only one crisp parameter, δ, to be updated. However, if we consider the fuzzy consequents of the rules, which are also, changed, then the total number of parameters is equal to (k+1). This is less than the number of parameters used in the neuro-fuzzy methods (Section 3.3.6).

#### *4.2.7 Fitting to Training Data*

The data fitting in the defuzzification-based algorithm is expected to be less precise than that in the considered neuro-fuzzy approaches. This is because the algorithm has a smaller number of parameters.

#### *4.2.8 Linguistic Interpretability*

The learning process described in Section 4.1 does not change the initial MF's assigned over the system inputs. Also, the consequents of the rules are selected from specified fuzzy sets over the output variable. Hence, with the input and output fuzzy sets assigned appropriately to permit a simple and clear linguistic labeling, then the generated rules will have a clear linguistic meaning. This serves well the issue of linguistic representation of knowledge but it is at the expense of accuracy as expected (See [11,12]). Data over-fitting, however, hinders the noise insensitivity and the generalization capability of the learning algorithm as shown in [9,14].

A 9-rule fuzzy system, with three triangular MF's on each input (as in Fig.6) and 7 triangular MF's on the output, was trained by the defuzzification-based algorithm. The 81 data noted in Section 3.3.8 were used. The final system had an error E=0.01234 and δ= 0.45.

#### *4.2.9 Firing State Problem*

Since the input MF's are not changed by learning, then unlike the considered neuro-fuzzy methods, the problem of non-firing states does not arise during or after learning.

## 5 Conclusion

This study has first provided a description and comparison between conventional and a new neuro-fuzzy system from the point of view of structure and learning-related properties. Both approaches require differentiable MF's, use fixed logic operations and error function, apply pattern learning and suffer from non-firing states during or after learning and from the lack of good linguistic interpretability. The new neuro-fuzzy approach, however, turned out to have a simpler setting of initial MF's and rules to avoid initial non-firing and smaller number of tuning parameters. Yet, the conventional approach has less complex learning formulas and more precise data fitting.

Then, a defuzzification-based algorithm has been summarized and shown to possess better properties than the considered neuro-fuzzy approaches. It does not require differentiable MF's nor fixed logic operations and error function. Setting the initial MF's and rules is even simpler than that in the new neuro-fuzzy method. The possibility of non-firing during of after learning is eliminated. Also, the algorithm provides completely interpretable Mamdani-type fuzzy systems with rules having fuzzy antecedents and consequents. Further, the algorithm employs batch learning, and its formulas apply easily to higher dimensional input spaces.

Although the algorithm provides less precise data fitting, this is not a disadvantage since it first provides fuzzy systems, which can easily be given a clear linguistic meaning. Besides, the available data in practice are noisy. This makes the reduced precision a needed aspect to improve noise insensitivity and generalization capabilities, as shown in [9,14]. Also, fuzzy modeling becomes more consistent with Zadeh's principle of "tolerance for imprecision" [13].

In fact, performance criteria related to noise insensitivity and generalization capabilities were introduced in [14] and the algorithm was examined and compared with ANFIS [15] using non-linear functions and a practical robot navigation case [14,16]. The performance advantages of the algorithm were demonstrated in these studies. Criteria-based performance comparison should also be done with the considered neuro-fuzzy approaches and also with an advanced method [17] accounting for noisy data.

## References

1. Ichihashi, H.: Iterative Fuzzy Modeling and a Hierarchical Network, Proc. 4th IFSA Congr., Brussels, (1992) 49 -52.
2. Ichihashi, H., Turksen, I.B.: A Neuro-Fuzzy Approach to Data Analysis of Pairwise Comparisons," Int. J. Approximate Reasoning, 9 (3), (1993) 227 -248.
3. Nomura, H., Hayashi, I., Wakami, N.: A Self-tuning Method of Fuzzy Control by Descent Method, Proc. IEEE Int. Conf. on Fuzzy Systems, San Diego, 1992, 203- 210.
4. Wang, L.X., Mendel, J.M. :Back-propagation Fuzzy System as Nonlinear Dynamic System Identifiers, Proc. IEEE Int. Conf. on Fuzzy Systems, San Diego, 1992, 1409-1416.
5. Shi, Y., Mizumoto M.: Some Considerations on Conventional Neuro-fuzzy Learning Algorithms by Gradient Descent Method, Fuzzy Sets and Systems, 112, (2000) 51-63.
6. Shi, Y., Mizumoto M.: A New Approach of Neuro-fuzzy Learning Algorithm for Tuning Fuzzy Rules, Fuzzy Sets and Systems, 112, (2000) 99-116.
7. Takagi, T., Sugeno, M.: Fuzzy Identification of Systems and its Application to Modeling and Control, IEEE Trans. Systems, Man and Cybernetics, 15(1), (1985) 116-132.
8. Saade,J.J.: New Algorithm for the Design of Mamdani-type Fuzzy Controllers, Proceedings of EUSFLAT-ESTYLF Joint Conference, Palma, Spain, Sept. 22-25, (1999) 55-58.
9. Saade, J.J.: A Defuzzification-based New Algorithm for the Design of Mamdani-type Fuzzy Controllers, Mathware and Soft Computing, 7, (2000) 159-173.
10. Saade, J.J.: A Unifying Approach to Defuzzification and Comparison of the Outputs of Fuzzy Controllers" IEEE Trans. Fuzzy Systems, 4 (3), (1996) 227-237.
11. Li, Y., Deng, J.M., Wei, M.Y.: Meaning and Precision of Adaptive Fuzzy Systems with Gaussian-type Membership Functions," Fuzzy Sets and Systems, 127, (2002) 85-97.
12. Paiva, R.P. ,Dourado, A.: Interpretability and Learning in Neuro-fuzzy Systems, Fuzzy Sets and Systems, 147, (2004) 17-38.
13. Zadeh, L.A.: Outline of a New Approach to the Analysis of Complex Systems and Decision Processes, IEEE Trans. Systems, Man and Cybernetics, 3(1), (1973) 28-44.
14 Saade, J.J., Al-Khatib, M.: Efficient Representation of Non-linear Functions by Fuzzy Controllers Design Algorithms, 7th IEEE International Conference on Electronics, Circuits and Systems, Dec. 17-19, (2000) 554-557.
15. Jang, J.S.R.: ANFIS: Adaptive-network-based Fuzzy Inference System, IEEE Trans. Systems, Man and Cybernetics, 23, (1993) 665-685.
16. Al-Khatib, M., Saade, J.J.: An Efficient Data-driven Fuzzy Approach to the Motion Planning Problem of a Mobile Robot, Fuzzy Sets and Systems, 134, (2003) 65-82.
17. Shi, Y., Mizumoto, M.: An Improvement of Neuro-fuzzy Learning Algorithm for Tuning Fuzzy Rules, Fuzzy Sets and Systems, 118, (2001) 339-350.

# Evolutionary Computation and Genetic Algorithms

# Intelligent genetic algorithm: a toy model application

Jaime Mora Vargas[1], Neil Hernández Gress[1], and Miguel González Mendoza[1]

Instituto Tecnológico y de Estudios Superiores de Monterrey,
Campus Estado de México, Carr. Lago de Guadalupe Km. 3.5
Atizapán de Zaragoza, Estado de México, México
{jmora, ngress, mgonza}@itesm.mx
http://www.cem.itesm.mx

**Abstract.** We argue that the performance of a genetic algorithm can be improved by the codification of its operative rates into the chromosome. In the case of the flowshop problem the claim is that mutation and crossover rates allow the genetic algorithm to adapt better and faster than the traditional genetic algorithm. We support our claim with a simple "toy model" with two instances of flowshop problem, an special case of scheduling with multiple applications to industrial problems. We refer to that genetic algorithm as Intelligent Genetic Algorithm (IGA) since its ability to self-modify its operative characteristics.

## 1 Introduction

Genetics algorithms have been applied to various optimization problems (Goldberg [5]). In this paper, a genetic algorithm is improved using local search procedures, and self-adaptation rates of genetic operators.

In the literature, many hybrid algorithms ([10, 4, 7, 14]) of GA's were proposed for flowshop optimization problems, those algorithms are a combination of traditional GA and artificial intelligence techniques (*e.g* tabu search, simulated annealing). In those studies, it was clearly shown that the performance of GA's for scheduling problems was improved using neighborhood search algorithms.

Flowshop problems are included into scheduling problems. Great efforts are devoted to its economical importance. Unfortunately, finding optimal scheduling for a general production process is an NP-hard problem (Garey and Johnson, [3]). This means that traditional operations research techniques such as integer programming (branch and bound techniques [13]) or dynamic programming (Bellman and Dreyfus [1]) are not adequate to deal with large scale problems. Therefore, the interest of many researchers has been oriented to find good solutions (not always a global optimum) in a reasonable time. Considering this, the use of metaheuristics techniques are well suited.

A major issue for metaheurists is the fine-tuning for parameters, *i.e* tabu list length (for tabu search), initial temperature (for simulated annealing) or crossover and mutation rates (for genetic algorithms). In this article, a modification is made to traditional GA, creating the "intelligent" Genetic Algorithm


*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 175-184*

(IGA) which doesn't need fine tuning of crossover and mutation rates. This technique was originally used with simple optimization landscapes ([17]) and to travelling salesman problem instances ([18]) with encouraging results.

This article is organized as follows: section 2 includes a general description of the flowshop problem as well as details for the 2 treated problems, also a brief introduction to GA is provided. Section 3 contains specific details about representation of flowshop problem, Section 4 includes details about the experiments done, Section 5 shows the most representative graphics and comments about experimentation and Section 6 has the conclusions and future work of this article.

## 2 Problem Description

Flowshop problems in particular, are a special case of scheduling problems and scheduling problems arise in the combinatorial optimization area. General assumptions for flowshop are (more details in Dudek *et al.* [2]): 1) jobs are to be processed by multiples stages sequentially, 2) there is one machine at each stage, 3) machines are available continuously, 4) a job is processed on one machine at a time without preemption, and 5) a machine processes no more than one job at a time. For this paper, $n$ jobs are processed in the same order on $m$ machines. Considering this, this paper works with a sequencing problem of flowshop scheduling of $n$-jobs.

Considering notation from Ishibuchi [8], the completion time and processing time of job $j$ on machine $i$ are $t_C(i,j)$ and $t_P(i,j)$ respectively. The $n$-dimensional vector $x = (x_1, x_2, \ldots, x_n)$ represents the sequence of $n$ jobs to be processed, where $x_k$ denotes the $k$-th processing job. Completion time for each sequence $x$ is calculated by:

$$t_C(1, x_1) = t_P(1, x_1) \tag{1}$$

$$t_C(i, x_1) = t_C(i-1, x_1) + t_P(i, x_1), i = 2, 3, \ldots, m, \tag{2}$$

$$t_C(i, x_k) = t_C(i-1, x_{k-1}) + t_P(i, x_k), k = 2, 3, \ldots, n, \tag{3}$$

$$\begin{aligned} t_C(i, x_k) = max\{t_C(i-1, x_k), t_C(i, x_{k-1})\} + t_P(i, x_k), \\ i = 2, 3, \ldots, m; k = 2, 3, \ldots, n \end{aligned} \tag{4}$$

Flowshop scheduling problems are to determine the sequence of $x$ of $n$ jobs based on a given scheduling criterion. According to Johnson's work [11] the reduction of makespan if one of the most extended criteria, also reduction of tardiness is employed. The makespan is the completion time of the last job:

$$Makespan(\mathbf{x}) = t_C(m, x_n) \tag{5}$$

Also maximum tardiness is other criteria used, it is defined as the maximum tardiness of the $n$ jobs to schedule, that is:

$$\begin{aligned} & Tardiness = \\ & max\{t_C(m,1) - d_1, t_C(m,2) - d_2, \ldots, t_C(m,n) - d_n\} \\ & t_C > d \end{aligned} \tag{6}$$

where $d_i$ represents due date for job $i$.

### 2.1 Genetic Algorithms

Genetic algorithms are one of the heuristic optimization algorithms widely used by many researchers in solving various problems, were introduced by Holland [6]. Genetic algorithms mimic the mechanism of genetic evolution in biological nature. In biological terms, it consist of a chromosome composed of genes, each one of them with several alleles, into the optimization field, this chromosome is a string that usually represents a possible solution to some optimization problem, each string is composed of bits with specific values. Initially, a number of chromosomes form an initial pool of solutions. The process of crossover and mutation will be carried out in the pool, after that an evolution is completed and new chromosomes (offspring) will be generated.

GAs have two major processes. First, GAs randomly generate new solutions. Second, the evolution of those initial solutions is done according to the genetic operators such as reproduction (selection of the fittest), mutation (exploration operator) and crossover (exploitation operator).

## 3 Problem Representation

### 3.1 Chromosome

Configuration for flowshop problem using GA uses a string base codification, where each individual in the population represents a possible sequence of jobs to be done. For example, the sequence $x = (1, 3, 2, 4)$ represents a sequence of 4 jobs, where job 1 is done first, followed by jobs 3, 2 and 4. That kind of representation is currently used to solve scheduling problems using GA.

### 3.2 Genetic operators

In this paper, two genetic operators were used : crossover and mutation in order to exploit results (crossover) and explore solutions (mutation). The crossover operator is the two-point order crossover and for mutation, it is used the shift change, details for such operators can be found in [14]. Such operators work selecting a random number and compare to the operation rates if it is smaller then the operator is applied to that individual.

In order to improve search for new solutions, a local search procedure was used, this procedure consisted in the permutation of size 1 for every population

element, selecting the best one. For example, for individual $(1, 2, 3)$, the possible neighbors would be $(2, 1, 3), (3, 2, 1)$ and $(1, 3, 2)$ this local search avoid the use of large populations, also it doesn't require important computational resources.

The reproduction of individuals is made using the so-called tournament reproduction of size $t$, where $t$=2, it function by selecting by random $N/t$ sets of $t$ elements and passing the element with the highest fitness of each set to the next generation, this procedure is done $t$ times to assure that the population number $N$ remains constant. Ties broken by random.

For the IGA, the standard genetic operators for binary codification [5] were used.

### 3.3 Fitness function

The fitness function used combines two objectives, minimize makespan and max tardiness. Using equations 5 and 6 it is possible to create a global equation for the $i$-th individual, that is:

$$f(i) = -log(MaxTardiness_i) - log(Makespan_i) \tag{7}$$

The *log* function is used in order to re-normalize the values of makespan and tardiness. As the GA nature is maximize, the use of "-" allows to get better results (*i.e.*, small makespan and small tardiness)

### 3.4 Intelligent GA

The "Intelligent Genetic algorithms" are a modification of traditional genetic algorithms in which the crossover and mutation rates are codified into the chromosome, for this paper, a string of 5 bits was used to codify in binary. In this manner, the max value (in decimal) is $2^5 = 32$ so it is possible to configure value rates between 0 and 1 with an interval of $1/32 = 0.03125$. The translation process consists in translate from binary to decimal and divide that value by the max value possible. Using this configuration, a complete individual is by example $[1, 2, 3, 4|00101|11000]$ representing that the first job to be processed is job 1, followed by jobs 2, 3 and 4, also the mutation probability is $00101 = 0.15625$ and the crossover rate is $11000 = 0.75$. Two types of genetics operators were used, those applied to the flowshop configuration and those applied to the operators rates configuration, for the flowshop configuration, two-point order (for crossover) and shift change (for mutation) were used. The traditional two parents, two points crossover and change between 0 and 1 mutation operator were used to the chromosome section that codifies operators rates. The sequence used was: first apply operators to flowshop section followed by the application of operators to codification rates section.

This self-codification allows the algorithm to avoid selecting optimal mutation and crossover rates, a time-consumption task that must be completed before run any standard GA. Special attention must be on IGA since self-adaptation capacity allows to apply GA into time-dependent landscapes.

## 4 Experiments

The experiments were realized using a flowshop problem of 5 machines-10 jobs (5M10J) and 5 machines-30 jobs (5M30J).

**Table 1.** Processing times 5M10J

| | j1 | j2 | j3 | j4 | j5 | j6 | j7 | j8 | j9 | j10 |
|---|---|---|---|---|---|---|---|---|---|---|
| m1 | 32 | 1 | 61 | 42 | 62 | 61 | 3 | 97 | 26 | 9 |
| m2 | 21 | 27 | 87 | 45 | 59 | 24 | 71 | 34 | 20 | 28 |
| m3 | 10 | 42 | 66 | 75 | 41 | 24 | 3 | 36 | 85 | 74 |
| m4 | 51 | 19 | 23 | 85 | 86 | 81 | 93 | 31 | 75 | 23 |
| m5 | 33 | 45 | 58 | 97 | 91 | 85 | 30 | 38 | 17 | 51 |

**Table 2.** Due date times, 5M10J

| job | Due date |
|---|---|
| j1 | 674 |
| j2 | 396 |
| j3 | 431 |
| j4 | 369 |
| j5 | 626 |
| j6 | 597 |
| j7 | 790 |
| j8 | 437 |
| j9 | 656 |
| j10 | 780 |

**Table 3.** Processing times 5M30J (jobs 1-10)

| | j1 | j2 | j3 | j4 | j5 | j6 | j7 | j8 | j9 | j10 |
|---|---|---|---|---|---|---|---|---|---|---|
| m1 | 32 | 1 | 61 | 42 | 62 | 61 | 3 | 97 | 26 | 9 |
| m2 | 21 | 27 | 87 | 45 | 59 | 24 | 71 | 34 | 20 | 28 |
| m3 | 10 | 42 | 66 | 75 | 41 | 24 | 3 | 36 | 85 | 74 |
| m4 | 51 | 19 | 23 | 85 | 86 | 81 | 93 | 31 | 75 | 23 |
| m5 | 33 | 45 | 58 | 97 | 91 | 85 | 30 | 38 | 17 | 51 |

Table 1 shows the processing times for job $n$ in machine $m$, for example, job 2 in machine 2 takes 27 time units, job 10 in machine 5 takes 51 time units. Table 2 shows the due date for each job. Tables 3,4 and 5 include the processing times for 5M30J, table 6 shows the due dates for 5M30J problem.

**Table 4.** Processing times 5M30J (jobs 11-20)

| | j11 | j12 | j13 | j14 | j15 | j16 | j17 | j18 | j19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| m1 | 47 | 5 | 35 | 88 | 84 | 40 | 79 | 94 | 56 | 13 |
| m2 | 29 | 35 | 81 | 94 | 77 | 30 | 19 | 75 | 47 | 30 |
| m3 | 43 | 19 | 49 | 85 | 79 | 55 | 34 | 93 | 64 | 50 |
| m4 | 25 | 30 | 83 | 80 | 83 | 32 | 45 | 88 | 49 | 62 |
| m5 | 50 | 40 | 85 | 78 | 77 | 49 | 80 | 48 | 44 | 20 |

**Table 5.** Processing times 5M30J (jobs 21-30)

| | j11 | j12 | j13 | j14 | j15 | j16 | j17 | j18 | j19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| m1 | 43 | 38 | 65 | 92 | 78 | 45 | 67 | 71 | 57 | 31 |
| m2 | 38 | 8 | 76 | 93 | 83 | 32 | 37 | 80 | 53 | 37 |
| m3 | 47 | 27 | 41 | 79 | 79 | 35 | 38 | 69 | 55 | 33 |
| m4 | 49 | 28 | 51 | 78 | 78 | 46 | 78 | 80 | 45 | 15 |
| m5 | 43 | 17 | 78 | 83 | 82 | 46 | 35 | 86 | 61 | 33 |

**Table 6.** Due date times, 5M10J

| job | Due date | job | Due date | job | Due date |
|---|---|---|---|---|---|
| 1 | 674 | 11 | 674 | 21 | 436 |
| 2 | 396 | 12 | 707 | 22 | 456 |
| 3 | 431 | 13 | 569 | 23 | 764 |
| 4 | 369 | 14 | 671 | 24 | 645 |
| 5 | 626 | 15 | 509 | 25 | 738 |
| 6 | 597 | 16 | 465 | 26 | 451 |
| 7 | 790 | 17 | 490 | 27 | 611 |
| 8 | 437 | 18 | 492 | 28 | 746 |
| 9 | 656 | 19 | 429 | 29 | 420 |
| 10 | 780 | 20 | 613 | 30 | 651 |

## 5 Results

The experiments carried out where done considering the problems mentioned in previous section, the objective of that experiments was to compare standard GA versus Intelligent GA. Both GA types used a population size of 500 individuals, and 510 generations. The provided results considers the average value for 20 runs per experiment.

For the standard genetic algorithms several experiments where done using different crossover and mutations rates. In this paper, results for fixed $mp$ (mutation probability) and $cp$ (crossover probability) are provided, the graphs shows results for $mp = 0.4$-$cp = 0.1$, $mp = 0.01$-$cp = 0.01$ which are compared with the IGA performance.

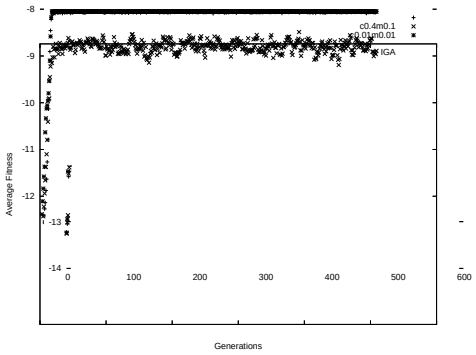



**Fig. 1.** Fitness comparison, 5 machines, 10 jobs.

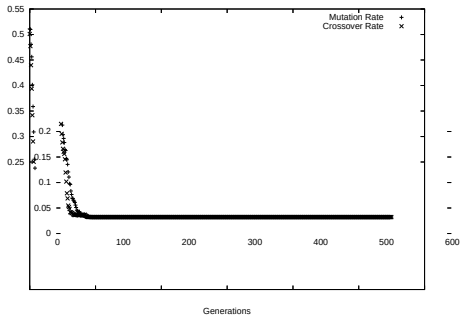



**Fig. 2.** Operative Rates, 5 machines, 10 jobs.

Figure 1 shows the comparison between average fitness for $mp = 0.4$-$cp = 0.1$, $mp = 0.01$-$cp = 0.01$ and IGA considering 5M10J problem, data showed are the average fitness for the entire population. For this case, $mp = 0.01$-$cp = 0.01$ and

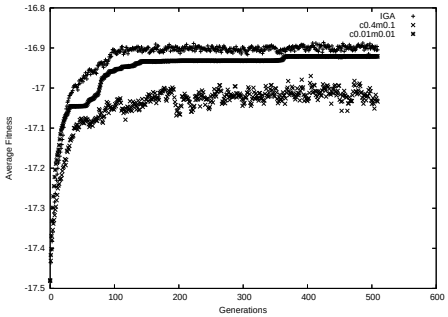



**Fig. 3.** Fitness comparison, 5 machines, 30 jobs.

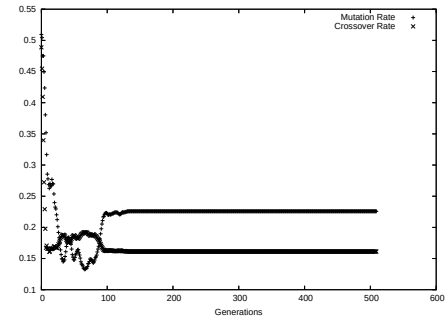



**Fig. 4.** Operative Rates, 5 machines, 30 jobs.

IGA have similar results. All three configurations find rapidly the optimal value (at generation 25 aprox). It is important to mention that 5M10J problem has a feasible space 10! = 3628800 size, which is very simple to solve using exhaustive-search procedures, the reason to use such landscape is to gain experience with the IGA and track their results.

The operative rates for genetic operators are showed in figure 2, one the IGA has find the optimum, it reduces its mutation and crossover rate, the latter achieving its stable value faster than mutation rate, this is related with the impact of the operator, *i.e.* mutation is a more destructive operator than crossover, then the search for new possible solutions continues by more time than the exploitation of results already found.

Figure 3 shows the comparison between $mp = 0.4$-$cp = 0.1$, $mp = 0.01$-$cp = 0.01$ and IGA considering 5M30J problem, data showed are the average fitness for the entire population. As this graph shows, it is clear that IGA have better performance than $mp = 0.01$-$cp = 0.01$. The convergence of IGA take more time than the others, the reason is that the IGA have to modified its operative ranges. Also, $mp = 0.01$-$cp = 0.01$ has a better performance than $mp = 0.4$-$cp = 0.1$, although $mp = 0.4$-$cp = 0.1$ goes faster to a local optimum, moreover $mp = 0.4$-$cp = 0.1$ has more changes between every generation this is because the mutation and crossover rates are relatively high, allowing to loose good solutions.

Figure 4 shows the mutation and crossover rate along the 510 generations of the experiment, note the changes in the rates, first descending to values of 0.15 for mutation and 0.16 for crossover. Again and similar to 5M10J mutation rate takes more values before get stable. Both crossover and mutation rates remains with the same value once an optimum is reached, and off course by the population effect (all individual have the same operation rates).

## 6 Conclusions

This article presents an application of a called "Intelligent Genetic Algorithms", a type of genetic algorithm which is able to modify its operational rates in order to achieve a global optimum. Such characteristic could be very important specially for problems in which the environment changes over time. Also IGA avoid fine-tuning of parameters, mostly always a time-consuming task.

The examples treated in the article are flowshop problems with 5 machines-10 jobs and 5 machines-30 jobs problems, in both examples treated, IGA have a better performance than standard GA, however it is possible to prove that by adjusting standard GA parameters it could perform better that IGA. Then the main application for IGA seems to be problems in which the environment changes over time, since the IGA can adapt to changes modifying its operative rates. In the case standard GA once the change occurs and since the majority of population is in the "old optimum" it can not be able to move to the new optima.

## References

1. Bellman, R.E. and Dreyfus, S.E., Applied Dynamic Programming, Princeton University Press (1962)
2. Dudek R.A., Panwalkar S.S. and Smith M.L., The Lessons of Flowshop Scheduling Search, Operations Research, Vol. 47 No. 1 (1992) 65-74
3. Garey, M. and Johnson, D.,Computers and intractability: A guide to the theory of NPCompleteness. Freeman and Co.San Francisco (1979)
4. Glass C. A., Potts C. N., and Shade P., Genetic algorithms and neighborhood search for scheduling unrelated parallel machines, Univ.Southampton, Southampton, U.K., Preprint Series OR47 (1992)
5. Goldberg, D.E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company. Reading, Massachusetts (1989)
6. Holland, J.H., Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor MI (1975)
7. Ishibuchi H., Yamamoto N., Murata T., and Tanaka H., Genetic algorithms and neighborhood search algorithms for fuzzy flowshop scheduling problems, Fuzzy Sets Syst., Vol. 67 (1994) 81-100
8. Ishibuchi H., Murata T. and Tomioka S., Effectiveness of Genetic Local Search Algorithms, Proc. 7th International Conference on Genetic Algorithms (1997) 505-520
9. Ishibuchi H. and Murata T., A Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling IEEE Transactions on systems, man, and cyberneticspart C: applications and reviews, Vol. 28 No. 3 (1998) 392-403
10. Jog P., Suh J. Y., and Gucht D. V., The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem, Proc. 3rd Int. Conf. Genetic Algorithms (1989) 110-115
11. Johnson S. M., Optimal Two- and Three-stage Production Schedules With Setup Times Included, Naval Research Logistics Quarterly, Vol. 1 No. 1 (1954) 61-68
12. Kursawe F., A variant of evolution strategies for vector optimization, Parallel Problem Solving from Nature, H.-P. Schwefel and R. Manner,Eds., Vol. 15 , Berlin Germany (1993) 754-770
13. Lawler, E.L. and Wood, D.E., Branch and Bounds Methods: A survey", Operations Research, Vol. 14 (1966)
14. Murata T. and Ishibuchi H., Performance evaluation of genetic algorithms for flowshop scheduling problems, Proc. 1st IEEE Int. Conf. Evolutionary Computat. (1994) 812-817
15. Murata T. and Ishibuchi H., MOGA: Multi-objective genetic algorithms, Proc. 2nd IEEE Int. Conf. Evolutionary Computat. (1995) 289-294
16. Schaffer J. D., Multi-objective optimization with vector evaluated genetic algorithms, Proc. 1st Int. Conf. Genetic Algorithms (1985) 93-100
17. Stephens C.R. and Mora J., Effective Fitness as an Alternative Paradigm for Evolutionary Computation I: General Formalism, Genetic Programming and Evolvable Machines, Vol. 1, No. 4 (2000) 363-378
18. Stephens C.R. and Mora J., Effective Fitness as an Alternative Paradigm for Evolutionary Computation II: Examples and Applications, Genetic Programming and Evolvable Machines, Vol. 2, No. 1 (2001) 7-32

# Improved Ant Colony System using Subpath Information for the Traveling Salesman Problem

Minyoung Yun and Inkyeom Kim

Department of Information and Communications Engineering,
Sungkyul University, Anyang, Korea
{alabama, kik}@sungkyul.edu

**Abstract.** Ant Colony System (ACS) applied to the traveling salesman problem (TSP) has demonstrated a good performance on the small TSP. However, in case of the large TSP, ACS does not yield the optimum solution. In order to overcome the drawback of the ACS for the large TSP, the present study employs the idea of subpath to give more information to ants by computing the distance of subpath with length 3. In dealing with the large TSP, the experimental results indicate that the proposed algorithm gives the solution much closer to the optimal solution than does the original ACS. In comparison with the original ACS, the present algorithm has substantially improved the performance. For a certain graph, the solution performance has been enhanced up to 72.7 % by utilizing the proposed algorithm.

## 1 Introduction

Ant System(AS) is a meta-heuristic algorithm proposed by Dorigo et al.[1] that has been inspired by the foraging behavior of ant colonies. Real ants are capable of finding the shortest path from a food source to their nest by exploiting pheromone information. Ant System was applied to the complex combinatorial optimization problems such as the traveling salesman problem (TSP) and the quadratic assignment problem (QAP). Currently many ongoing research activities has been performed to investigate many different discrete optimization problems like vehicle routing, sequential ordering, graph coloring, and routing in communication networks.

In the present study, the Ant Colony System has improved the efficiency of the existing ant system and it has been applied to analyze TSP. In context with the Ant Colony System, the ants acting like agents perform parallel search for the TSP and find a good solution. During this process, the ants are able to exchange information each other indirectly but globally by using pheromone [5]. Each ant constructs the path for TSP with the iterative procedure to select a next visiting city by jointly utilizing informations on the greedy heuristic and the past experience. Several meta-heuristic search algorithm are applied to find an optimal solution for TSP which is well known as a NP-hard problem.

The TSP can be expressed by a complete weighted graph $G = (V, E)$. Here $V$ is a set of vertices and $|V| = n$, it represents all cities that the sales person has to


*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 185-194*

visit. The $E$ denotes a set of edges. Each edge $(i, j) \in E$ has a weight $d_{ij}$ which represents a distance between any two cities $i$ and $j$ $(i, j \in V)$. Consequently, the TSP can be converted to a Hamiltonian circuit problem which find a shortest path from a starting city by visiting each city only once and returning to the starting city on a complete weighted graph. The TSP is classified as symmetric TSP and asymmetric TSP. In the asymmetric TSP, the distance of the paired vertices $(i, j)$, $d_{ij}$, could be different for the circulating direction. In other word, there exists at least one edge which satisfies $d_{ij} \neq d_{ji}$. In the symmetric TSP, $d_{ij} = d_{ji}$ is satisfied for every edges in $E$.

The original ACS algorithm is capable of finding an optimal solution for the small size of TSP. The original ACS uses information on distance of adjacent neighbors only. However, in case of the large TSP, ACS does not yield the optimum solution. In order to overcome the drawback of the ACS for the large TSP, the present study employs the idea of subpath to give more information to ants by computing the distance of all possible subpath with length to construct a tour for a solution. In dealing with the large TSP, the experimental results indicate that the proposed algorithm gives the solution much closer to the optimal solution than does the original ACS. For a certain graph, the solution performance has been enhanced up to 72.9 % by utilizing the proposed algorithm. In comparison with the original ACS, the present algorithm has considerably improved the performance. The detailed discussion has been made for the existing and proposed algorithm for the ant colony optimization to solve the large TSP with a symmetry.

## 2 Ant Colony Optimization Algorithms

The Ant Colony Optimization(ACO) algorithm is easily applicable to handle the TSP. In the ACO algorithm, the pheromone trails consist of the connecting edges and $\tau_{ij}$ represents the measure of possibility to visit a city $j$ directly from a city $i$. The heuristic information is expressed as $\eta_{ij} = 1/d_{ij}$. The values of $\tau_{ij}$ and $\eta_{ij}$ are stored at pheromone matrix and heuristic matrix, respectively. For each ant, tours are constructed by the following procedure : (1) choose a start city in random fashion and place an ant; (2) according to values of $\tau_{ij}$ and $\eta_{ij}$, construct a path by adding a city that the ant has not visited yet; and (3) after all cities have been visited, go back to the starting city and complete one path. After all ants have completed their tour, they may deposit a certain amount of pheromone according to the tour they have constructed [7,8].

The Ant System(AS) is a initially developed ACO algorithm and it is quite easy to apply the TSP. However, due to the simple pheromone updating rule, there is a certain tendency the AS leads to the local optima situation. Therefore, the AS gives the optimal solution only for the small TSP. To improve the performance, several extensions of the AS was devised. These extensions include elitist AS, rank-based AS, and MAX-MIN AS. The main difference between the original AS and these extensions is the way to update the pheromone[4]. The ACS algorithm is the framework of this study and it is the ACO algorithm by adopt-

ing the basic idea of the AS. Its performance has been improved by overcoming the drawbacks of the AS. The ACS has been applied to various combinatorial optimization problems and it has demonstrated a good performance.

The ACS proposed by Gambardella and Dorigo[9] differs from the AS in the following features:

1. By using a more aggressive action choice rule, compared to the AS, the ACS more actively exploits the search informations accumulated by the ants.
2. Pheromone evaporation and pheromone deposit take place only on the edges belonging to the best-so-far tour.
3. Each time an ant uses an edge $(i, j)$ to move from city $i$ to city $j$ , it removes some pheromone from the edge to increase the room for selecting the alternative paths.

In the initial stage of the ACS with a given graph $G = (V, E)$ and $|V| = n$, $m$ ants ($m \leq n$) are placed on $m$ cities in random fashion. According to the tour construction rule, each ant repeatedly chooses a next visiting city and constructs a path. In this process, whenever an edge is added to a path, the local pheromone updating rule is applied to update the pheromone on each edge. When the path is constructed, the local search is applied to improve the constructed path. Then the pheromone is updated only at the global optimal path with the minimum length among all paths constructed so far. Figure 1 shows the ACS algorithm for the TSP.

```
algorithm: ACS for TSP {

   Initialize Data;
   while (not terminate) {
      place m ants at m cities;
      repeat (for each ant)
           apply tour construction rule to build a trail;
           apply local pheromone updating rule;
      until (construct a solution)
      apply local search;
      apply global pheromone updating rule;
   }
}
```

**Fig. 1.** Algorithm: ACS for TSP

### 2.1 Tour Construction Rule

If ant $k$ is located at city $i$, then a next visiting city $j$ can be chosen according to the pseudo-random proportional rule, given by equation (1).

$$j = \begin{cases} \arg \max_{l \in N_i^k} \{\tau_{il}[\eta_{il}]^\beta\}, & \text{if } q \leq q_0 \\ J, & \text{otherwise} \end{cases} \tag{1}$$

where $\beta$ is a parameter which determines the relative importance of pheromone $\tau_{ij}$ versus heuristic information $\eta_{ij}$, ${N_i}^k$ is the set of the remaining cities to be visited by ant $k$ positioned on city $i$. $q$ is a random variable uniformly distributed in $[0, 1]$, $q_0$ is a parameter to satisfy the range, $0 \leq q_0 \leq 1$ , and $J$ is a random variable selected by the following probability distribution.

$$p_{ij}^k = \frac{[\tau_{ij}][\eta_{ij}]^\beta}{\sum_{l \in N_i^k}[\tau_{il}][\eta_{il}]^\beta} \qquad \text{if} \quad j \in N_i^k \tag{2}$$

In the equation (2), the probability to select an edge $(i, j)$ in a path is dependent on the amount of pheromone, $\tau_{ij}$ and heuristic information, $\eta_{ij}$. Each ant select a city $j$ as a next visiting city which has a large level of pheromone and a short distance. If $\beta = 0$, the selection of a next city depends only on the pheromone level, $\tau_{ij}$. Therefore, in the general situations, $\beta > 1$, according to reference[4], a good performance is achieved at $2 \leq \beta \leq 5$.

### 2.2 Local Pheromone Trail Update

Unlike the AS, the ACS uses a local pheromone updating rule. Whenever an ant constructs a tour of the TSP and select an edge, the pheromone level for a selected edge is updated by applying the local updating rule equation (3).

$$\tau_{ij} = (1 - \xi)\tau_{ij} + \xi\tau_0 \tag{3}$$

where $\xi$ is the variable to satisfy the range, $0 \leq \xi \leq 1$. According to numerical experiment, the best performance is achieved at $\xi$ =0.1 [4]. The value of $\tau_0$ represents the initial pheromone level and the best performance is obtained at $\tau_0 = 1/(nC^{mn})$, where $n$ is the number of cities in the TSP and $C^{mn}$ is the length of a path constructed by the nearest-neighbor heuristic. In other word, the pheromone level at each edge is initialized by the length of a path which is constructed by the greedy method. By applying the equation (3), whenever an ant selects an edge $(i, j)$, its pheromone level, $\tau_{ij}$ at a selected edge is reduced. As a result, the once selected edge has the much lower probability to be selected by the following ants. This treatment increases the probability to select the edges that have not been visited yet and it prevents from a stagnation behaviour which is a certain tendency to repeatedly choose an once selected edge. In other words, ants do not converge to the generation of a common path. In this study, we only consider symmetric TSP such that $\tau_{ij} = \tau_{ji}$.

### 2.3 Local Search

The local search is basically included in the ACO algorithm. After all ants have completed to find their own path, the locally optimum solution can be obtained by 2-opt or 3-opt procedure which exchange two or three edges involved in the constructed path. If this local search is applied to construct a path of the TSP, the ACO algorithm together with a local search can improve the solution constructed by an ant[10]. In this proposed algorithm, a 3-opt method is employed.

### 2.4 Global Pheromone Trail Update

In the procedure of a Global Pheromone Trail Update, the pheromone update is allowed only for the most optimum path among all constructed paths, according to the equation (4).

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \qquad \forall(i,j) \in T^{bs} \tag{4}$$

where $\Delta\tau_{ij}^{bs}$ is the amount of pheromone to be added to edge $(i,j)$ in the optimal path. $C^{bs}$ represents the length of global optimal solution. Thus, the relation between the pheromone level and the optimum path length is expresses as $\Delta\tau_{ij}^{bs} = 1/C^{bs}$ . The parameter $\rho$ is the pheromone evaporation rate. The deposited pheromone is decreased with increasing the pheromone evaporation rate, $\rho$. In experiments, the best performance is obtained at $\rho = 0.1$.

## 3 Proposed Algorithm

The ACS algorithm adopts a global pheromone update as well as a local pheromone update. If the global pheromone update is used, the information about the best path among all constructed paths is delivered to ants which start to search for the solution. On the other hand, the local pheromone update decreases the pheromone on the edge which is just visited by ants. Therefore, this procedure increases the probability to select the edges that have not been visited yet and it can avoid a stagnation behaviour and increase a room to find a optimum path. However, in case of a graph having a large number of vertices, it is difficult to find the optimal path by only using the heuristic method.

In the ACS, $m$ ants are placed randomly on $m$ cities and start to search for the optimal path. During the searching process using the local pheromone updating rule, if the current visiting city is highly probable to be included in an optimal path, the probability to find an optimum path is definitely increased by constructing the path based on the current visiting city.

In the present study, to give the more precise information to ants for constructing a optimum path, the value of $\eta_{ij}$ in equation (1) is not determined just by using the distance of adjacent neighbor and it is determined by using the subpath $s_w$ with length $w$, $1 \leq w \leq n$. Here $w$ refers the number of edges including subpath. Using the information about the length of subpath $s_w$, we precompute the length of all city $i$ based subpaths which can be constructed from $w$ number

of edges, $(i,j)(j,k)...(x,t)$. Then the next visiting city is selected as a city which is located at the subpath with the minimum length among all possible subpaths. In other words, in the searching process of a next visiting city $j$ from the city $i$. the tour path is constructed by selecting a city which minimize the value of $s_w$. According to this algorithm, the equation (1) can be modified as follows:

$$j = \begin{cases} \arg \ \max_{l \in N_i^k} \{\tau_{il}[\eta_{il}{}^{sw}]^{\beta}\}, & \text{if } q \leq q_0 \\ J, & \text{otherwise} \end{cases} \tag{5}$$

Through numerical procedure of this algorithm, we need to make the list of the nearest neighbor first, and then we have to find the minimum distance between the neighboring cities in the list. For instance, if we assume the subpath length $w = 3$, we first compute the distance of subpath $(i,j)(j,k)(k,t)$, $d_{ij} + d_{jk} + d_{kt}$ for every city $j$ adjacent to the current city $i$. Then the nearest neighbor list, $l_i$ is arranged by sorting with an ascending order. As illustrated in the Figure 2, this procedure marginally increases the total execution time owing to its preprocessing treatment. As implied in the local pheromone updating method represented by equation (3), it is quite important how to evaluate a initial value of pheromone, $\tau_0$ because it continuously influences the tour construction process. In the original ACS, a initial value of pheromone is obtained by $\tau_0 = 1/(nC^{mn})$. Here $C^{mn}$ is the length of path which is constructed by the Greedy method. On the other hand, in the present proposed algorithm, a initial value of pheromone is evaluated by the following expression (6) which has a governing parameter, $C^{sw}$.

$$\tau_0 = 1/(nC^{sw}) \tag{6}$$

Since $C^{sw}$ is generally smaller than $C^{mn}$, every path in this proposed algorithm has the much higher level of initial pheromone. Moreover, using equation (3) together with a initial value of pheromone, $\tau_0$ governed by the subpath information $C^{sw}$, the present proposed algorithm can search the adjacent neighbors more precisely.

If the original ACS is applied to the TSP with large number of cities, it is very difficult to find an optimal solution. In the search procedure of optimal solution for the TSP, if the correct and optimal algorithm is applied, the shortest path is quickly found at the beginning of the search process. Otherwise, the optimum path could be constructed by gradually improving the solution through the numerous iterations. However, in case of a large TSP, it is nearly impossible to construct a optimal path by applying the iterative search for all possible paths. Therefore, it is necessary to include cities in a path which are quite probable to construct the optimal path. By setting the initial value of pheromone large, the proposed algorithm constructs the path much closer to a optimal solution at the initial stage than does the original ACS algorithm. In other words, the proposed algorithm can increase a probability to find an optimal path at the beginning of path construction stage by choosing a nearest neighbor with the subpath information. However, it is necessary to note that, if the length of subpath, $w$,

is set to a quite large value, it is susceptible to be a local minima. If the length of subpath, $w$, is set to a small value, then there is no difference with original ACS and the algorithm performance is greatly reduced. Therefore, it is very important to set $s_w$ with the proper value. Figure 2 shows the schematics of proposed algorithm. Here the italictype parts represent the major improvements against the original ACS.

```
algorithm: Proposed ACS for TSP {

  preprocessing steps:
      construct a distance matrix;
      construct a nearest neighbor list by s_w;
  Initialize Data;
  while (not terminate) {
      compute τ_0 with s_w : τ_0 = 1/(nC^sw);
      place m ants at m cities;
      repeat (for each ant)
          apply tour construction rule to build a trail;
          apply local pheromone updating rule;
      until (construct a solution)
      apply 3-opt local search;
      apply global pheromone updating rule;
  }
}
```

**Fig. 2.** Proposed Algorithm

## 4 Experimental Results and Discussion

The proposed algorithm has been implemented into the aco-code in reference [11]. For the validation, we used the graphs in the TSPLIB library [12]. The experiments on the proposed algorithm have been performed at Enterprise RedHat 2.1 (PentiumIV 1.7 GHz, 768MB). For each test, we have chosen the parameters which were proved to yield the optimal solution from the previous experiments. These problem parameters are given as $\xi = 0.1, \rho = 0.1, \beta = 2, q_0 = 0.9$ and $m = 10$. The initial value of pheromone in the equation (6) is evaluated by using $\tau_0 = 1/(nC^{s3})$ and the information of subpath is obtained from $s_3$. For each ant, 100 seconds of CPU time are allocated for one search process and the path search is repeated 10 times. For each graph, the optimum and averaged value is obtained from the results of 10 executions.

Table 1 shows the results obtained from the original ACS and the proposed algorithm for the graphs with less than 1000 cities for the length of subpath 3, $s_3$. Here, 'Instance' represents the graph name in TSPLIB and 'Known optimal' represents the known length of the optimal path for the corresponding

graph. The 'Best' and 'Average' of original ACS denote the optimal and averaged lengths calculated by the Dorigo's algorithm[8]. On the other hand, the 'Best' and 'Average' of proposed ACS corresponds to the optimal and averaged lengths computed by the proposed algorithm. The 'NNChangeRate' in the last column represents the changing rate in the next visiting city which is determined by the proposed algorithm with $s_3$, versus to the original ACS. As shown in the experimental results, in case of the graphs with small number of cities, the original and proposed algorithm can find the optimal solution within the fairly short period.

**Table 1.** Experimental results for the graphs with less than 1000 cities

| *Instance* | *Known* | *Original ACS* | | *Proposed ACS* | | *NNChange* |
|---|---|---|---|---|---|---|
| | Optimal | Best | Average | Best | Average | Rate(%) |
| att 532 | 27686 | 27686 | 27704.28 | 27686 | 27705.88 | 36.47 |
| d 198 | 15780 | 15780 | 15780.19 | 15780 | 15780.1 | 23.23 |
| lin 318 | 42029 | 42029 | 42086.48 | 42029 | 42087.58 | 19.18 |
| pcb 442 | 50778 | 50778 | 50835.83 | 50778 | 50831.57 | 13.57 |
| rat 783 | 8806 | 8806 | 8819.88 | 8806 | 8821.01 | 24.65 |
| d 1291 | 50801 | 50801 | 50874.87 | 50801 | 50863.21 | 7.20 |

In case of the graph att532, experimental results obtained by the proposed algorithm indicate that the changing rate of the next visiting city is more than 35%, compared to the original ACS. This situation can be occurred when the graph has the more complexity and the large number of edges. Since the generated subpaths in this complex graph situation are rapidly increased, the possibility to change the next visiting city becomes higher. On the other hand, in case of the graph d1291 having more than 1000 cities and simple edge connection among cities, the NNChangeRate is only 7% because the probability to change the next visiting city becomes lower for the simple graph situation.

However, in case of the graph with more than 1000 cities and high complexity, it is quite seldom to find an optimal solution by employing the original ACS. Table 2 illustrates the experimental results of the graphs with more than 1000 cities. As shown in Table 2, in case of the large graph, the proposed algorithm finds the solution much closer to the optimal solution than does the original ACS. The 'Improved Rate' at the rightmost column represents the improvement rate of the searching path constructed by the proposed algorithm, compared to the original ACS. This improvement rate is evaluated by the relation, 100 - {(c-a)/(b-a) * 100}. Experimental results simulated by the proposed algorithm indicate that the only 0.5% improvement is obtained for the graph, rl 1889 and the 72.7% improvement is for the graph, r1 5915. Even if the improvement rate has a certain level of sensitivity for the specific graph, experimental results for most of the large graphs show that the proposed algorithm yields more than 30% of improvement. These experimental results suggest that, in dealing with

the large and complex graph, the proposed algorithm is much better than the original ACS in terms of efficiency and performance improvement.

**Table 2.** Experimental results of the graphs with more than 1000 cities

| *Instance* | *Known* | *Original ACS* | | *Proposed ACS* | | *NNChange* | *Improved* |
|---|---|---|---|---|---|---|---|
| | Optimal(a) | Best(b) | Average | Best(c) | Average | Rate(%) | Rate(%) |
| d 1655 | 62128 | 62153 | 62357.89 | 62147 | 62352.75 | 11.0 | 34.0 |
| fnl 4461 | 182566 | 186492 | 186986.05 | 186361 | 187032.61 | 29.75 | 3.4 |
| pcb 3038 | 137694 | 139098 | 139749.38 | 138933 | 139661.64 | 26.37 | 21.8 |
| rl 1889 | 316536 | 317349 | 319232.81 | 317345 | 318849.68 | 20.12 | 0.5 |
| rl 5915 | 565530 | 576654 | 581050.39 | 575837 | 581286.21 | 21.05 | 72.7 |
| u 1432 | 152970 | 153204 | 153579.93 | 153131 | 153612.97 | 1.47 | 31.2 |
| vm 1748 | 336556 | 336765 | 337531.19 | 336679 | 337641.23 | 25.69 | 41.2 |
| pr 2392 | 378032 | 378838 | 380344.11 | 378654 | 380418.47 | 20.03 | 32.8 |

In general, the proposed algorithm shows a good performance for the most of the graphs. However, as shown in Table 2, the performance is still sensitive to the characteristics of each graph. Since the original ACS basically adopts the greedy heuristic algorithm to search for an nearest neighbor, the search process to find a optimal path could be highly influenced by the distance from the nearest neighbor. Thus, the performance of the ACS algorithm could be improved by changing the value of parameters according to the size of graph or number of edges in the graph. In case of the graph u1432, NNChangeRate is just 1.47% but the solution obtained by the proposed algorithm is up to 31.2%. This result implies that, in this particular graph, the performance can be significantly improved by changing few cities in visiting order. In contrast to the graph u1432, the graph fnl 4461 is another extreme case. In case of the graph fnl 4461, NNChangeRate is nearly 30% and the improved rate is only 3.4%. Since a changing rate of the nearest neighbor list is quite high according to the information on subpath $w_3$, it can be speculated that a graph fnl 4461 could have the much higher complexity. Therefore, in this type of a complex graph, any meta-heuristic algorithms may yield the similar trend for the solution of TSP. The experimental results suggest that the proposed ACS algorithm could be improved by varying the subpath length, $w$ according to the characteristics of graphs.

## 5 Conclusion

In this study, we propose an algorithm which improve the performance of the original ACS for the TSP. For the construction of tour, the original ACS search the adjacent cities first, then select a city with the minimum distance as the next visiting city. However, in order to optimally choose the next visiting city, the proposed algorithm uses the information on subpath such that the distance

of all possible subpaths with length $w$ are precomputed and select a city having the much higher probability to construct a optimal path. If the length of subpath, $w$ is long, there is a possibility for stagnation. Therefore, it is quite crucial to select the proper subpath length, $w$. In the ACS, the information on subpath $s_w$ highly influences the initial value of pheromone, $\tau_0$. Since the value of $\tau_0$ is continuously used in the updating process of local pheromone, it eventually influences the tour construction process.

In case of the graphs with small number of cities, the original and proposed algorithm can find the optimal solution within the fairly short period. For the large TSP, with the same CPU time, the proposed algorithm finds the solution much closer to the optimal solution than does the original ACS. Even if the improvement rate has a certain level of sensitivity for the specific graph, experimental results for most of the large graphs show that the proposed algorithm enhances the improved rate more than 30%. For a certain graph, the solution performance has been improved up to 72.7% by utilizing the proposed algorithm. The experimental results suggest that the proposed ACS algorithm could be improved by varying the subpath length, $w$ according to the characteristics of graphs.

## References

1. M. Dorigo, V. Maniezzo and A. Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26(1) : 1-13, 1996.
2. M. Dorigo and L. M. Gambardella. Ant Colonies for the Travelling Salesman Problem. *BioSystems*, 43:73-81, 1997.
3. S. Lee and T. Chung. A Study about Additional Reinforcement in Local Updating and Global Updating for Efficient Path Search in Ant Colony System. *Journal of Korean Information Processing*, 10(B):237-242, 2003.
4. M. Dorigo, G. D. Caro and L. M. Gambardella. Ant Algorithms for Discrete Optimization. *Artificial Life*, 5(3):137-172, 1999.
5. O. Gomez and B. Baran. Reasons of ACO's Success in TSP. *Proceedings of 4th International Workshop in Ant Colony Optimization and Swarm Intelligence*, LNCS 3172: 226-237, 2004.
6. L. M. Gambardella and M. Dorigo. Solving symmetric and asymmetric TSPs by ant colonies. *Proceedings of IEEE International Conference on Evolutionary Computation*, IEEE-EC 96: 622-627, 1996.
7. M. Dorigo and T. Stutzle. Ant Colony Optimization. MIT Press, 2003.
8. E. Bonabeau, M. Dorigo and G. Theraulaz. Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, 1999.
9. M. Dorigo and L. M. Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53-66, 1997.
10. L. Bianchi, J. Knowles and N. Bowler. Local search the probabilistic salesman problem: correction to the 2-p-opt and 1-shift algorithms. Technical Reports, IDSIA-18-03, Dalle Molle Institute of Artificial Intelligent, Switzerland, 2003.
11. http://www.aco-metaheuristic.org/aco-code/
12. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/

# Building Block Filtering Genetic Algorithm

Jun Lu, Boqin Feng and Bo Li

Xi'an Jiaotong University

**Abstract.** A Building Block Filtering Genetic Algorithm(bbf-GA) is proposed which introduces building block candidates filtering and exploiting to improve traditional GA**.** Various recognition functions are designed and tested by analyzing the features of building blocks during the evolution of GA search for symmetrical TSP, and one of them is adopted to filter building block candidates. A position representation for TSP and relevant bbf-based genetic operators are designed to exploit the building block candidates. The proposed TSP specialized position representation can decrease the computational workload of bbf-GA, such as edge comparison, computation of individual similarity, abstraction of uniform edge, and operations in bbf-based genetic operators. Experimental results show that comparing with traditional GA, Building Block Filtering Genetic Algorithm can improve the efficiency of search remarkably by reducing unnecessary search in GA.

## 1 Introduction

The GA's search strategy is commonly described by the pattern theorem and building block hypothesis. The building block hypothesis (Holland1975; Goldbery 1989) [1] states that the GA works well when short, low-order, highly-fit schemas recombine to form even more highly fit higher-order schemas. The ability to produce fitter and fitter partial solutions by combining building blocks is believed to be the primary source of the GA's search power, thus improving the ability of GA to exploit known building blocks in limited populations and to explore new building blocks at the same time is essential to improve the search of GA.

Numerous researchers have studied on defining and exploiting building block. Forest and Mitchell[2] designed a class of fitness landscapes ( the "Royal Road" function) to measure the effects of genetic operators on building block in binary encoding mechanism. Wu *et al.* [3] compared two different GA representation schemes with a floating representation scheme and examine the differences in building block dynamics. Kemenade[4] compared and identified building block by calculating the difference of the fitness value caused by the change of allele in binary coding, and utilized it in the proposed three-stage GA.

Zhou peng *et al*. [5] applied reduction mechanism to find uniform partial solutions from local optimal solutions generated by heuristic method, then reduced the scale of the instance by multi-reduction algorithm, finally the solution of the original problem could be reverted after iterative operations. Schneider *et al.* [6] proposed an efficient parallel method to reduce the instance of TSP to a smaller one by finding backbones which are actual uniform partial solutions from local optimal solutions and eliminating them from original problem to get even better solutions in a very short time and a few observables of interest corresponding to this parallel approach.

These research works inspire us to find the way to guide the search of GA by filtering building block from similar parts among populations. In this paper, the search of

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 195-206*

instance of symmetrical Traveling Salesman Problem (TSP) is used to evaluate various recognition mechanisms, and one of them is adopted and a Building Block Filtering Genetic Algorithm (bbf-GA) is proposed. The structure of this paper is as follows. In next Section we introduce the chromosome representation for TSP (position representation) that we proposed, as well as compare and analyse various recogniztion functions and their filtering results. After that, the bbf-GA is described detailed in Section 3 and experimental results are presented in Sections 4. Finally, conclusions come in Section 5.

## 2 Recognition of Building Block

The Traveling Salesman Problem has been in the focus of studies for many years. In order to investigate the features of building block during the typical evaluation of a GA search, several testing instances in TSPLIB[7] are chosen.

As to TSP, Building blocks can be taken as the "best" edges. For a certain node, the "best" edge is not the shortest edge that takes it as vertex (greedy algorithm can hardly find the optimal solution), but the edge that synthesized with other "best" edge to make the shortest full-path. Thus, it is impossible to determine an edge is good or not just by the comparison of edges' length. Calculating the difference of the full-path length when the connected edge of a vertex is changing could identify best edges. However, the computational work of this is as hard as that of the solution search itself.

In the first place, we use traditional GA to solve TSP. The known optimal solution is inputted at the beginning of algorithm and all edges in the solution are taken as "best" edges, which are the building blocks. Then we look into the distribution and change of building blocks in each individual during running process of algorithm to find the way to filter building blocks. In order to improve the calculating efficiency, position representation is designed.

### 2.1 Position Representation

There have been many different representations used to solve the TSP problem using GA [8] such as ordinal representation(Grefenstette 1985), adjacency representation (Grefenstette 1985), metrix representation(Fox and McMahon 1992), edge representation (Homaifar and Guan 1993), and path representation etc. The most natural representation is path representation. For instance, path (1-3-2-6-5-4-1) can be represented as (1,3,2,6,5,4) directly. However in this representation, the individual has a cycle topology. The meaning of genic segment just shows the relationship between a node and its previous and next node, but is independent of its position in chromosome. Different individual, such as the four shown in the left column of Table 1, may represent the same path.

In algorithms based on path representation, it takes much time in recognizing the same edges in two individuals. Thus, we propose a position representation inspired by adjacency representation (Grefenstette 1985). In position representation, each individual is composed of two parts: right adjacency (RA) and left adjacency (LA), which

means the subsequence node and previous node of the node that represented by genic position (in adjacency representation individuals only have right adjacency). E.g. path(1-3-2-6-5-4-1)can be represented as(RA)(3 6 2 1 4 5)and(LA)(4 3 1 5 6 2) where the 3rd position in(RA)is 2 which means edge (3-2), and the 2nd position in(LA) is 3 which also means edge (3-2).

The position representation of the four individuals is depicted in the right column of Table 1. For individuals in position representation, it only needs two operations to judge whether an edge in them is the same or not. E.g. the following comparison is used to judge whether edge (4-5) in individual 2 exists in individual 3 or not.

if(individual2.RA[4]==individual3.RA[4] || individual2.RA[4]==individual3.LA[4])

It's easy to find from Table 1 that six edges of those four individuals are all the same. Although position representation requires more memory, it reduces the computational work for the comparison of allele among individuals. What is more, it benefits bbf-based genetic operators depicted in Section 3.

**Table 1.** The comparison of path representation and position representation

| position / individual | path representation | | | | | | position representation | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 1 | 3 | 2 | 6 | 5 | 4 | 3 | 6 | 2 | 1 | 4 | 5 |
| | | | | | | | 4 | 3 | 1 | 5 | 6 | 2 |
| 2 | 1 | 4 | 5 | 6 | 2 | 3 | 4 | 3 | 1 | 5 | 6 | 2 |
| | | | | | | | 3 | 6 | 2 | 1 | 4 | 5 |
| 3 | 2 | 6 | 5 | 4 | 1 | 3 | 3 | 6 | 2 | 1 | 4 | 5 |
| | | | | | | | 4 | 3 | 1 | 5 | 6 | 2 |
| 4 | 6 | 2 | 3 | 1 | 4 | 5 | 4 | 3 | 1 | 5 | 6 | 2 |
| | | | | | | | 3 | 6 | 2 | 1 | 4 | 5 |

## 2.2 Recognition function of building block

Firstly, Simple Genetic Algorithm (SGA) is used to investigate the distribution of building blocks in the evolution of a GA search for Ludwig's drilling problem 280. Unfortunately, the results in early stage are depressed. When the population size is set to 400, only 30 edges are as same as those in the optimal solution even iterate to the $1000^{th}$ generation. In order to reduce runtime, new individuals in each generation are optimized by 2-opt algorithm in probability Ph (that is so-called memetic algorithm). 2-opt algorithm can eliminate path crossover effectively, but cannot guarantee to find optimal solution [8]. The amount of building blocks among the population in the $20^{th}$ generation during the evolution of a search is shown in Fig. 1 (where x-axis is the serial number of buiding blocks, which are the edges in the optimal solution and y-axis is the relevant appearing frequency in the population). We can see that some building blocks have a quite high appearing frequency at the beginning of evolution. Thus, it is possible to recognize building block by certain statistic methods to avoid

useless and repeatable search for known building blocks. In order to find the relationship between the amount of building blocks among individuals and the length of the path of individuals, we analyze these two features of individuals. The results are shown in Fig. 2.

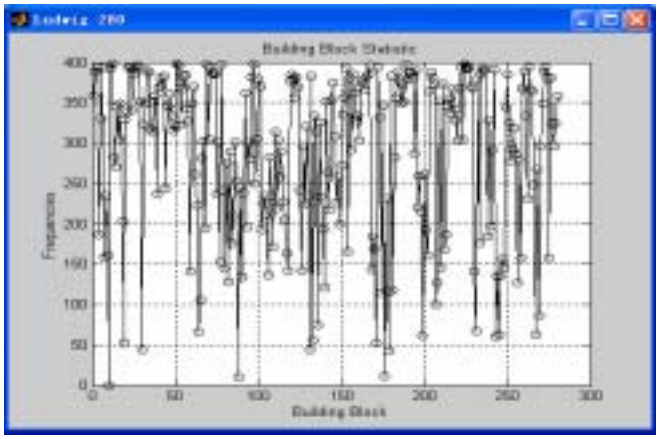


**Fig. 1.** Statistic of appearing frequencies for building blocks

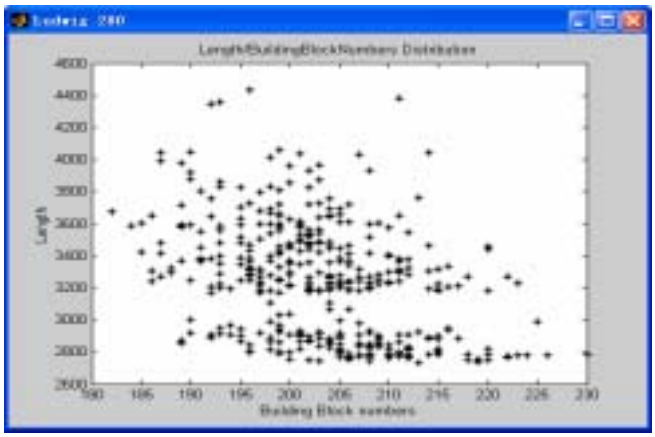


**Fig. 2.** Distribution of path length / build blocks in individuls

From Fig. 2 we can see that the amounts of building blocks among individuals in populations are ranged from 180 to 230, and the amount of building blocks and the path length do not have linear relationship. The individuals that have longer path length may include more building blocks on the contrary. As to TSP problem, an individual that has a worse edge achieves a longer path length no matter how many good edges it has.

Six statistic functions are designed to recognize building block from populations, in addition the recognizing effects are evaluated.

Let N be the size of population, $f_i$ be the fitness of individual (fitness function is $f(\pi)=76.5\times L\sqrt{N}/D\pi$ ,where $L$ means the side of the smallest square which can contain all the cities, $N$ is the number of cities and $D\pi$ is the length of the path in the current permutation.), $\bar{f}$ be the average of fitness, n be the amount of individuals whose fitness are better than $\bar{f}$ , $E_i$ be the set of edges in individual i, $E_i^e=\begin{cases}1 & e\in Ei\\ 0 & e\notin Ei\end{cases}$ ,then six recognition functions are designed as follows:

$$F_e^1=\sum_{i=1}^{N}E_i^e/\mathrm{N} \tag{1}$$

$$F_e^2=\sum_{i}^{n}E_i^e/\mathrm{n} \tag{2}$$

$$F_e^3=\sum_{i}^{N}E_i^e*(\frac{fi}{\sum_{i}^{N}fi}) \tag{3}$$

$$F_e^4=\sum_{i}^{n}E_i^e*(\frac{fi}{\sum_{i}^{n}fi}) \tag{4}$$

$$F_e^5=\sum_{i}^{n}E_i^e*c(1-c)^{i-1}\qquad c\in[0.01,0.05] \tag{5}$$

$$F_e^6=\sum_{i}^{b}E_i^e/\mathrm{b}\qquad \mathrm{b}\in[2,\ \mathrm{N}] \tag{6}$$

### 2.3 Analysis of recognition functions

Probabilities are calculated by recognition functions for all edges in population in the generations during the evolution. Those edges whose probability exceed threshold $P_{threshold}$ are considered as building block candidates, and let $T_C$ represents the amount of them. Compare building block candidates and building blocks (edges in known optimal solution) to find false building blocks (that are candidates who are not true building blocks), and let $F_C$ represents the amount of these false building blocks. Then, the true recognition rate is: Rate=$(T_c-F_c)/T_c$.

When $P_{threshold}$ is set to 0.98,the comparison of the recognition results of function $F^1$ to $F^5$ is shown in Fig. 3.

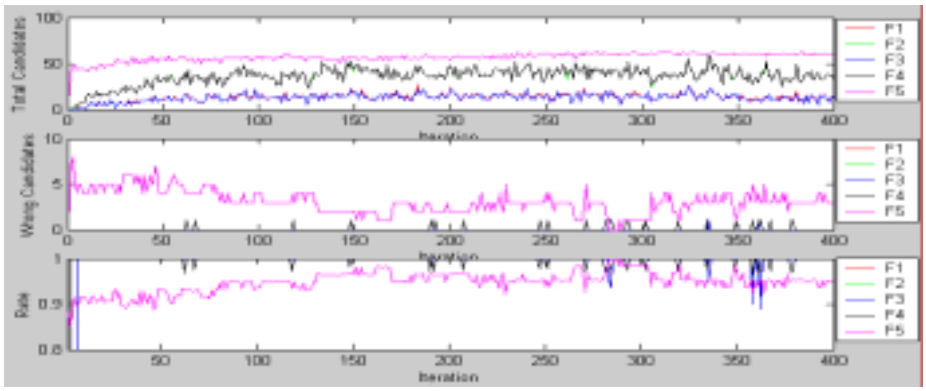



**Fig. 3.** The comparison of different functions ($P_{threshold}$ =0.98)

By analyzing a great deal of experimental data, we found that the recognition ability of function $F^4$ is the best, while $F^5$ is the worst whose false rate is the highest. The false rate of $F^3$ is the sub-highest, while the rest are similar. The comparison of recognition results with different thresholds of function $F^4$ is shown in Fig. 4.

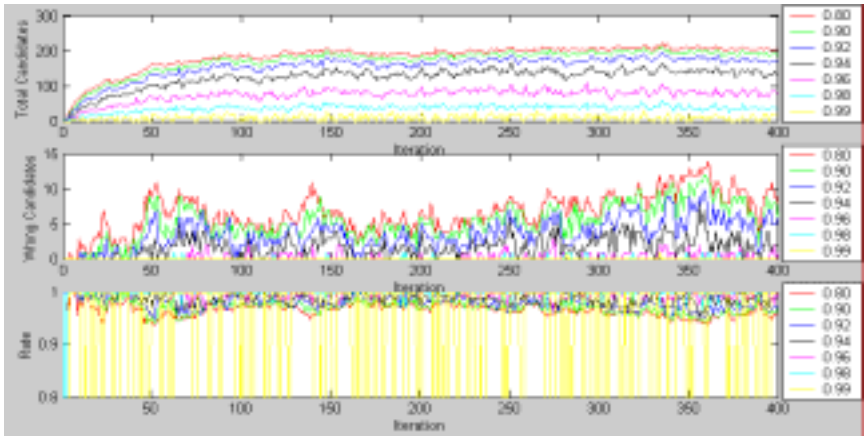



**Fig. 4.** Comparison of different thresholds for $F^4$ ($P_{threshold}$ is ranged from 0.80 to 0.99)

From Fig. 4 we can see that when the threshold is close to 1, it is hardly to find building block candidates, and when the threshold is lower than 0.96, the false rate is rather high.

The recognition results of function $F^6$ are shown in Fig. 5. When b, which means the number of the best individuals in population, is set from $2^1$ to $2^7$, and the threshold is 1. From Fig. 5 we can see that the smaller the number of statistic individuals is, the higher the false rate is. When b is set to 8 and at the 190$^{th}$ generation, although the true rate is 0.95, due to the larger number of building block candidates, the false building blocks are over 20. From the comparison of all mechanisms, we find that function $F^4$($P_{threshold}$ =0.98) and $F^6$($P_{threshold}$ =1 , b= $2^7$) , the comparison of which is

shown in Fig. 6, are better than others. As a result, we take $F^6$ ($P_{threshold}$ =1 , b= $2^7$) as filtering function for that it needs less computational work.

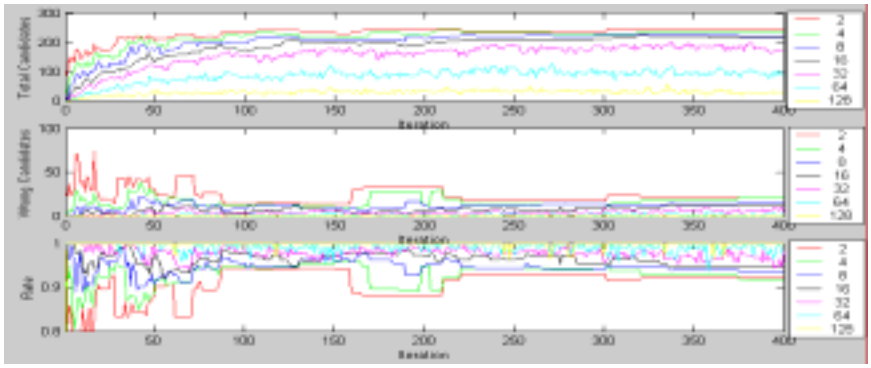


**Fig. 5.** Comparison of $F^6$ ($P_{threshold}$ =1) when b is set from $2^1$ to $2^7$

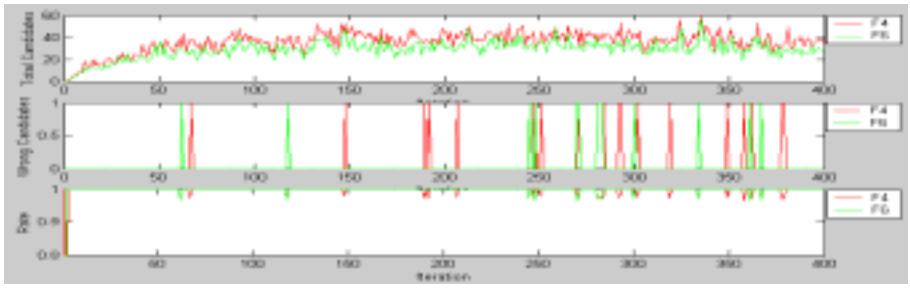


**Fig. 6.** Comparison of $F^4$ ($P_{threshold}$ =0.98) and $F^6$($P_{threshold}$ =1, b=$2^7$)

The comparison of different functions show that statistic based method can recognize building block in a high probability. But false recognition will appear no matter which function is used. In this case, eliminating the edges of building block candidates from original problem by reduction mechanism to reduce the scale of the problem will probably cause false reduction, which will result to failure to find the optimal solution.

However, the average probability of finding building blocks by the best functions is 0.98. These building block candidates can be preserved during the evolution and make the search of GA more effective.

# 3 Building Block Filtering Genetic Algorithm (bbf-GA)

## 3.1 The bbf-GA

In order to exploit the building blocks filtered, we propose bbf-GA as following:

```
Initial the parameters of GA;
Create initial population P(t) randomly;
Improve chromosomes by 2-opt algorithm in probability Ph;
Evaluate P(t);
While (not meeting the terminal condition){
    Calculate and abstract building block candidates from individuals of P(t);
    Implement crossover operation to P(t) in probability of Pc*(1-Pb) to get C1(t);
    Implement bbf-based crossover operation to P(t) in probability of Pc*Pb to get C2(t);
    Reproduce P(t) in probability of (1-Pc) to get C3(t);
    C(t) =C1(t)+C2(t)+C3(t);
    Implement mutation operation to C(t) in probability of Pm*(1-Pb);
    Implement bbf-based mutation operation to C(t) in probability of Pm*Pb;
    Implement 2-opt algorithm to C(t) in probability of Ph;
    Evaluate C(t);
    Generate P(t+1) based on the optimum individuals in P(t) and C(t);
    t=t+1;
}
```

In our algorithm the position representation is adopted, and each individual is represented as right adjacency (RA) and left adjacency (LA). Crossover operator adopts Ordered Crossover Operator[8] method proposed by Davis in 1985, which constructs an offspring by choosing a sub tour of one parent and preserving the relative order of cities of the other parent. Mutation operator adopts random multipoint mutation. The parameters of GA are: Pc(crossover probability), Pm(mutation probability), Ph(2-opt optimization probability) , N(the size of population).

In order to filter building block candidates, the recognition function is implemented to algorithm. Building block candidates are also represented as right adjacency (RA) and left adjacency (LA), where the position of non-building block is represented as -1. In addition, bbf-based crossover operator and mutation operator is designed to exploit building block candidates. The bbf-based genetic operators are used in probability of Pb, while normal operators are used in 1-Pb. From later experiments we can see that the search will cause rapid premature convergence when Pb is big enough. Due to the existence of false genic segment in building block candidates, Pb shouldn't be too big. When Pb is set to 0, algorithm is equal to traditional GA actually.

### 3.2 Bbf-based genetic operators

In this paper, traditional genetic operators are mended to exploit building block candidates in individuals. The bbf-based crossover operator is depicted as following.

*Input:* parents P1,P2; building block candidates B

*output:* offspring O1

*operation:* choosing edges in parent P2 that either are between random position s1 to s2 or belong to building block candidates and the rest edges from parent P1 to generate offspring O1. The loss edges are generated randomly.

Algorithm description:

```
(1)  iCount=0;              // count of passed node
(2)  O1[ ]=φ                // path of offspring
(3)  Set edges in P2 that neither are between position s1 to s2 nor belong to building
     block candidates to -1;
(4)  iCur=rand(N);   // begin with random node
(5)  while(iCount <N){   // analyze for each gene
(6)       if(P2.RA[iCur]!=-1){   / /P2 has right adjacency
(7)            iNext= P2.RA[iCur];
(8)            P2.RA[iCur]=-1;            // segment can be used only once
(9)            P2.LA[iNext]=-1;
(10)      }
(11)      else if(P2.LA [iCur]!=-1){   //P2 has left adjacency
(12)           iNext= P2.LA[iCur];
(13)           P2.LA[iCur]=-1;            //segment can be used only once
(14)           P2.RA[iNext]=-1;
(15)      }
(16)      else{       // edges that not belong to P2 are selected from P1
(17)           iNext= P1.RA[iCur];                  // select RA first
(18)           if(iNext∈O1[ ] || (P2.RA [iNext]!=-1 && P2.LA[iNext]!=-1) ){
(19)           // next node has been used, or is the vertex of two edges in P2
(20)                iNext= P1.LA[iCur];             // select LA then
(21)                if(iNext∈O1[ ] || (P2.RA[iNext]!=-1 && P2.LA[iNext]!=-1) )
(22)                // next node has been used, or is the vertex of two edges in P2
(23)                iNext=random usable node
(24)           }
(25)      }
(26)      O1[ ]+=iNext;
(27)      iCur=iNext;
(28) }
```

The purpose of step 3 in above algorithm is to eliminate neither selected edges nor building block candidates in parent P2 to generate offspring O1 cooperated with parent P1. The starting node is generated randomly, and next node is selected from P1 or P2 each time. To generate next node, the RA and LA (which means two edges representing different direction from current node) of parent P2 are selected firstly. If no usable node found, the RA and LA of parent P1 is used instead. If no valid node is

found in both P1 and P2, next node is generated randomly (in step 23). In this case, it is necessary to estimate whether next node is used (which may cause cycle), or is the vertex of two edges in P2 (which may cause the loss of an edge in P2).

The algorithm preserves all edges that either in selected zone or belong to building block candidates in parent P2, as well as some edges of parent P1. From the process of algorithm we can see the advantage of position representation, that is, preserving certain edges (non-continuous edges are possible) effectively without much computational work.

The way to mend mutation operator is simple. After generating a node that should be mutated randomly, the RA and LA of this node are checked whether they are belonging to building block candidates. If so, generate a new node to avoid losing of these edges.

The bbf-based operators can preserve edges that belong to building block candidates in parents and avoid damage, reform and comparison to these nodes. Thus, the search is focus on the rest edges, which reduce unnecessary stochastic search and improve search efficiency of GA.

## 4 Experiment and Analysis

In our experiments, we set parameter values as followings: population size N=400, number of generations=400, crossover probability Pc=0.80, mutation probability Pm=0.03, local optimization (2-opt algorithm) probability ph=0.3. Traditional GA and bbf-GA (Pb=0.35, and bbf-based genetic operators are employed after $100^{th}$ generation) are implemented 100 times each and the results are shown in Table 2.

**Table 2.** Comparison of Traditional GA and bbf- GA

| Algorithm | Times of finding optimal solution | Average generations of finding optimal solution | variance of solution |
|---|---|---|---|
| Traditional GA | 4 | 309 | 69.14505583 |
| bbf-GA | 28 | 213 | 29.08194754 |

From Table 2 we can see that the possibility of finding the optimal solution (2586.7696) by bbf-GA is increased remarkably, and the fluctuating of result is reduced. Another important data in our experiment is that a suboptimal solution (2587.8088) is found 48 times. We consider this as a result of that the false building block candidates cause convergence to suboptimal in high probability. The results also show that traditional GA has higher fluctuating and randomicity than bbf-GA. Fig. 7 is the distribution of building block candidates in one of these results (the red edges represent building block candidates filtered).

**Fig. 7**. Distribution of building block candidates

The bbf-GA cannot be improved by increasing Pb simply. When Pb is up to 0.85, the result is no better than traditional GA. By analyzing the recognizing process of building block candidates we can see that when Pb is big enough, building block candidates (including both true and false candidates) diffuse among population. It is clear to see from Fig. 8 that the amount of building block candidates and false candidates are increased significantly as well as recognition accuracy is reduced from the $100^{th}$ generation when bbf-based genetic operators are employed.

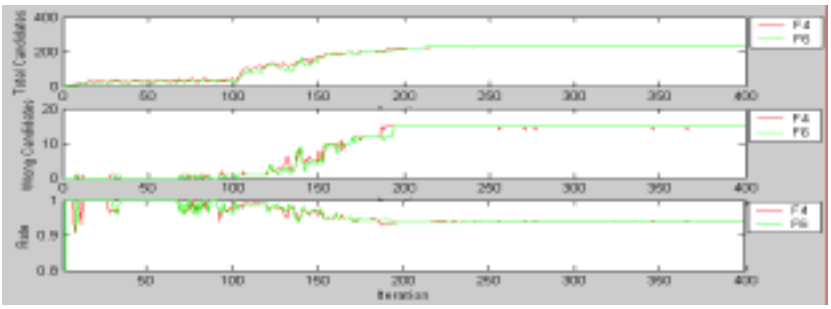


**Fig. 8.** Comparison of candidate building block recognition in evaluation process (Pb=0.85)

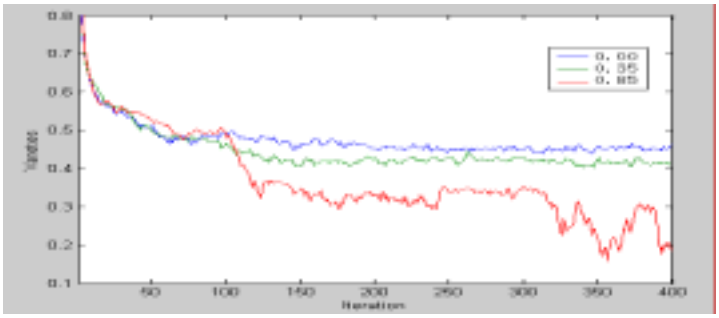


**Fig. 9.** Diversity comparison in evoluation process

By observing the individuals among population, we find that when Pb is big enough, the diversity of population will be destroyed, which leads to premature convergence. Fig. 9 shows the comparison of the average information entropy of different generations, where when Pb=0 (traditional GA), the diversity is the highest; when Pb=0.35, the diversity is decreasing slightly; When Pb=0.85, the diversity is decreased significantly from the $100^{th}$ generation.

## 5 Conclusion

In order to reduce useless search of GA on parts that are already optimal and make the search more effective, a mechanism that uses statistic function to filter building block candidates in the evolution of GA search is proposed. By testing the recognition effect of 6 statistic functions, a bbf-GA is proposed, including the filtering of building blocks and the bbf-based genetic operators. The experimental results show that the recognition and utilization of building blocks can improve the efficiency of search significantly. The comparison between traditional GA and bbf-GA makes it clearly that local searching algorithm(2-opt) can generate a large amount of high quality partial solutions rapidly, as well as recognizing and preserving these partial solutions during the evolution of GA can take advantage of the parallel search ability of GA. In addition, position representation is proposed, which decreases the computational workload of bbf-GA, such as edge comparison, computation of individual similarity, abstraction of uniform edge, and operations in bbf-based genetic operators (especially for the exploitation of non-continuous edges).

## References

1. J. H. Holland. Adaptation in Natural and Artificial System. Ann Arbor: The University of Michigan Press, 1975.
2. S. Forrest, M. Mitchell. Relative building-block fitness and the building-block hypothesis. In Foundations of Genetic Algorithms 2, 1992, pp.109-126.
3. Annie S. Wu and Kenneth A. De Jong An examination of building block dynamics in different representations. In the Proceedings of the 1999 Congress on Evolutionary Computation, 1999 pp. 715-721.
4. Cees H. M. van Kemenade. Building block filtering and mixing. Report SEN //Centrum voor Wiskunde en Informatica, Software Engineering, 1998.
5. Zhou Peng, Zhou Zhi, Chen Guoliang et al. A multilevel Reduction Algorithm to TSP. Journal of Software.2003,14(1):35-42.
6. J.Schneider, et al, Searching for backbones-an efficient parallel algorithm for the traveling salesman problem, Computer Physics Communications 96, 1996, pp.173-188.
7. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/
8. P.Larranaga, C.M.H.Kuijpers, et al. Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review, 13(2), 1999, pp. 129-170.

# Evolutionary Training of SVM for Classification Problems with Self-Adaptive Parameters

Ángel Kuri-Morales[1], Iván Mejía-Guevara[2]

[1] Departamento de Computación, Instituto Tecnológico Autónomo de México,
Río Hondo No. 1,
01000 D. F., México
akuri@itam.mx
[2] Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, IIMAS, Ciudad Universitaria,
04510 D. F., México
imejia@uxmcc2.iimas.unam.mx

**Abstract.** In this paper we describe a methodology to train Support Vector Machines (SVM) where the regularization parameter (C) is determined automatically via an efficient Genetic Algorithm (Vasconcelos' GA or VGA) in order to solve classification problems. We call the kind of SVMs where C is determined automatically from the application of a GA a "Genetic SVM" or GSVM. In order to test the performance of our GSVM, we solved a representative set of problems. In all of these the algorithm displayed a very good performance. The relevance of the problem, the algorithm, the experiments and the results obtained are discussed.

## 1 Introduction

Support Vector Machines have recently received increasing attention from the scientific community due to their underlying mathematical foundation. As opposed to more informal (and traditional) alternatives to neural network development, SVMs rely on well understood mathematical properties which, in effect, allow us to theoretically prove that, for example, perceptron networks (PN) or radial basis function (RBF) ensembles are all encompassed by them. Architectural issues such as the number of hidden layers and the number of neurons in such layers are dispensed with. A number of parameters the user has to heuristically estimate (such as the learning rate in PNs or the number of centers in RBFs) are not present. One key issue in this kind of networks, however, has to do with the so-called "regularization parameter" which, in effect, determines the accuracy of the SVM in terms of possible misclassification of sample elements unknown during the training phase. "C", as of today, has been traditionally determined on a case basis and, although some prior efforts to automate its value do exist [1] there has not been a reliable report of it systematic case-independent automated calculation. In this paper we propose the use of evolutionary computation techniques which help us solve the problem of C's determination; particularly, we focus on classification problems. In Section 2 we discuss some theoretical issues re-

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 207-216*

garding SVMs, specifically emphasizing the importance of regularization parameter C. In section 3 we discuss how the methodology of VGA can be used to train this kind of NN and show how to determine automatically the regularization parameter from its application to the dual problem. We also argue that this methodology is appropriate to solve constrained optimization problems, such as these. In section 4 we present four problems we analyzed to show how the GSVM may solve Classification Problems and the resulting level of accuracy. Three of these data sets were obtained from the University of California at Irvine Machine Learning Repository (UCI-MLR); a fourth was derived theoretically. In section 5 we discuss the experiments and results. Finally, in Section 6 we offer our conclusions and point to future lines of research.

## 2 Support Vector Machines

SVM is a supervised neural network that has been used successfully for classification and nonlinear regression problems [2][3][4]. In what follows we use the notation "$x_i$" and "w" to denote the independent variable vectors and the weight vectors respectively. A training sample $\tau = \{(x_i, d_i)\}_{i=1}^{N}$ (where $x_i$ is the input pattern for the *i*th example and $d_i$ is the target output) represents two classes in the case of pattern classification and a set of *N* independent variables with *N* dependent variable $(d_i)$ in the case of nonlinear regression.

When attempting pattern classification, the objective is to find a surface that allows the separation of the objects in the sample in two classes: the first class should be on one side of the surface $(d_i = 1)$ and the second class on the other side $(d_i = -1)$. The distance between the nearest points of both classes is called the margin of separation and the optimal surface is found when that margin is maximized.

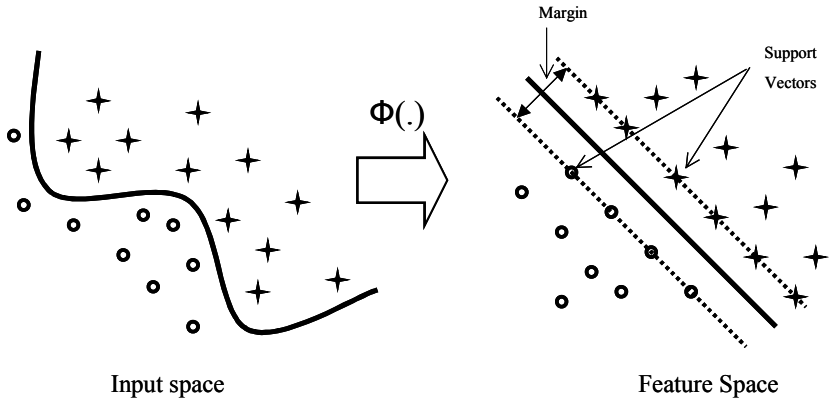



**Fig. 1.** Transformation from input space to higher-dimensional feature space.

The form of the surface depends of the linear separability characteristics of $\tau$, i. e., when $\tau$ is "linearly separable" the optimal surface corresponds to a hyperplane that is

called "Optimal Hyperplane" (OHP) and when $\tau$ is "nonlinearly separable", the optimal surface is not a hyperplane in the input space. The introduction of kernel functions is made in order to deal with non-linear decision surfaces. This implies mapping the data to a higher dimensional feature space which allows the construction of an OHP in this space that adequately separates the two classes. In Figure 1, the class 1 (squares) and the class 2 (stars) are non-linearly separable in the input space. In the feature space, however, both classes are separated from each other with a hyperplane.

The kernel functions are used to map vectors in the input space into vectors in the feature space. These functions must satisfy certain known conditions to be admissible as kernels in a SVM. Specifically they must satisfy Mercer´s condition [5][6]. Many functions may be used as kernels [7], but the most popular are: a) Polynomial learning machines (PLM), b) Radial-basis function networks (RBF) and c) Two-layer perceptron networks (LP) [8]. Since the theory allows for any of the above, we used PLM and RBF kernels due to their proven simplicity.

### 2.1 Primal and dual forms

As mentioned above, we want to find the OHP which maximizes the margin of separation between the two classes that constitute the training set. This gives rise to a constrained optimization problem which has to be solved to get the OHP. The form of the problem depends on linearly separable characteristics of the training set. The Quadratic Programming (QP) problem for linearly separable patterns is formulated as follows:

$$\begin{aligned} &\underset{w,b}{Min}\ \Phi = \frac{1}{2} w^T w \\ &\text{subject to :} \\ &d_i\left(w^T x_i + b\right) \geq 1 \quad \text{for } i = 1, 2, ..., N \end{aligned} \tag{1}$$

The solution of this problem requires the search of $w$ and $b$ that minimize an objective convex function subject to a set of linear constrains. In the case of nonlinear patterns, a set of slack variables is introduced $\{\xi_i\}_{i=1}^{N}$ in order to control the level of misclassification for some elements of $\tau$ [9]. In this case the QP problem is:

$$\begin{aligned} &\underset{w,b,\xi}{Min}\ \Phi = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \\ &\text{subject to :} \\ &d_i\left(w^T x_i + b\right) \geq 1 - \xi_i \quad \text{for } i = 1, 2, ..., N \end{aligned} \tag{2}$$

Equations (1) and (2) correspond to primal problems for classification of linearly and nonlinearly separable classes, respectively. However, it is possible to define the dual problem. The optimal value for both problems is the same [10]. In both problems, (1) and (2), the solution of the dual form corresponds with the Lagrange Multipliers (LMs) of the QP problem and the LMs different from zero correspond to the support vectors [11]. The dual form for nonlinearly separable patterns is:

$$Max\, Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j K(x_i, x_j)$$
$$\text{subject to:} \quad \sum_{i=1}^{N} \alpha_i d_i = 0 \quad 0 \le \alpha_i \le C \quad \text{for } i = 1, 2, ..., N \tag{3}$$

The dual form for separable patterns is essentially the same, except for $\alpha_i, i = 1,..., N$ which are not bounded; here, C is the upper bound on $\alpha_i$. It is important to note that a kernel is included in the dual form ($K(\cdot,\cdot)$), a fact which permits us to construct a decision surface that is nonlinear in the input space but whose image in the feature space is linear.

**Regularization Parameter.** The upper bound C for the LMs in a nonlinearly separable QP problem is known as "Regularization Parameter" [12]. This parameter controls the trade-off between the complexity of the machine and the level of misclassification allowed. When C is low, a higher proportion of errors is allowed in the solution, while few errors are permissible for high C values.

**Automatic determination of C via GA.** "C" is traditionally selected by the user. It may be estimated experimentally or analytically [13]. The analytical option relies on the calculation of Vapnik-Chervonenkis (VC) dimension for the problem at hand. VC dimension is, however, extremely difficult to calculate in practice and, in effect, disallows the analytical approach. Therefore, the main goal of this paper is to propose a method to estimate automatically the optimal value of this parameter using a GA without the practical limitations mentioned above. In our approach C's value is in the genome and induces a new constraint. This possibility is exclusive of the evolutionary approach (and perhaps a few other meta-heuristics) and explains our choice.

## 3 Genetic Algorithms

GAs are nowadays commonly used to solve complex optimization problems [14]. It is natural to tackle the problem of finding a good value of "C" with one. In what follows we briefly discuss the methodology.

### 3.1 Training a SVM using GAs

Several commercial optimization libraries can be used to solve the quadratic programming problem. However, these libraries are of limited used. The memory requirements of the QP problem grow with the square of the size of the training sample [15]. For that reason, in real-life applications, the QP problem cannot be solved by straight forward use of a commercial optimization library. Some optimization tech-

niques can be directly applied to QP problems. However, many of them require that the kernel matrix is stored in memory, implying that the space complexity is quadratic in the sample size. For large size problems, these approaches can be inefficient, and should therefore be used in conjunction with other techniques [16]. In this paper, we use GAs to tackle the QP problem.

**GAs as optimization tool**. The application of GAs to SVMs differs substantially from previous approaches to train NNs because the dual QP problem presented above is used to find the support vectors directly. In previous experiences the support vectors have been determined from the application of Lagrange Multipliers which neatly adjust to this problem (which satisfies Karush-Kuhn-Tucker conditions) but which are not applicable to search for "C" [13]. In fact, GAs are used here to solve the constrained QP. One advantage of using GAs for this kind of problems is that restrictions are not imposed in the form of the objective function: neither the objective function nor the constrains of the problem must be derivable in order to solve de problem properly.

### 3.2 Relative optimality of VGA

Although GAs were originally designed to solve unconstrained optimization problems, they can be adapted to tackle the constrained cases [17] as will be shown.

The first step is the selection of the population's size. In this work we considered a population of size P = 100 for all of the problems; the initial population was randomly generated; weighted binary fixed point representation was used. Each individual represents a LM ($\alpha_i$, i=1,...,N), where N is the number of points in the training set for the dual SVM problem. Every variable is to be expressed in fixed point format with one sign bit (0→+; 1→-), 8 integer bit and 20 decimal bits as shown in figure 2.

| $\alpha_i$ | | |
|---|---|---|
| **Sign** | **Int** | **Dec** |
| 1 bit | 8 bits | 20 bit |

**Fig. 2.** Fixed point representation

With this representation: $-2^8+2^{-20} \le \alpha_i \le +2^8-2^{-20}$. The genome's size is (N+1)x29, where N is the number of training data (N) and the (N+1)th point corresponds to the value of C. Once the initial population is generated, VGA [18] is used with $P_m$=0.05 (probability of mutation) and $P_c$=0.9 (probability of crossover). The evaluation function is constructed following the methodology of SVMs but we modify it by transforming the constrained original problem to a non-constrained one. To do this, we have chosen the penalty function (*F(x)*) [19]:

$$F(x) = \begin{cases} \left[ Z - \sum_{i=1}^{s} \frac{Z}{t} \right] - f(x) & s \neq t \\ 0 & otherwise \end{cases} \quad (4)$$

where $Z$ is a large constant [$O(10^9)$], $t$ is the number of constraints, $s$ is the number of these which have been satisfied and $f(x)$ corresponds to the fitness at point $x$. The GA operation was terminated after 250 generations.

## 4 Training SVMs for classification problems

We have applied the methodology to the problem of determining whether an object from a sample belongs to one of two groups. This may be easily extended to N groups [20]. SVMs have traditionally been designed to deal with binary classification, but a lot of real world problems have more than two classes. In this paper we deal with both, binary and multi-class problems. In the case of multiple class problems, one-versus-one classifier and one-versus-all classifier [21] were used. In one-versus-one classifier, a SVM model is built for each pair of classes. This results in $p(p-1)/2$ ($p$ is the number of classes in a specific problem) SVM classifiers. In one-versus-all classifier, $p$ classifiers are used. The ratio between the number of classifiers in one-versus-one classifier and one-versus-all classifier is $(p-1)/2$, which is significant when $p$ is large. On the other hand, all $N$ observations are used in each classifier in one-versus-all classifier.

### 4.1 Problems

A set of classification problems is presented here in order to illustrate the classification efficiency of the method. The set of problems are:

**Lung Cancer Database.** The data for this problem describes 3 types of lung cancers. The Authors give no information on the individual variables nor on where the data was originally used[1]. A total of 32 instances are considered in the original data. Since it has 5 missing attributes only 27 were considered. The data have 56 predictive nominal (values 0-3) attributes. Three classes are considered in this problem with: 9 observations for class 1, 13 for class 2 and 10 for class 3. It is important to mention that the problem has few instances (27) and a lot of attributes (55). For this reason we decided to use natural splines [22] to interpolate and enrich the data. The new (interpolated) data set consisted of 100 objects: 85 were used for training and 15 for testing.

**Wine Recognition Database.** These data are the result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultures[2]. It corresponds to three types of wines with a total of 178 instances: 59 for wine class 1, 71 for class 2 and 48 for class 3. A total of 13 continuous attributes for each object was considered.

[1] UCI-MLR [http://www.ics.uci.edu/~mlearn/MLRepository.html]
[2] Idem

**Iris Plant Database.** This is perhaps the best known database to be found in the pattern recognition literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant[3]. One class is linearly separable from the other two, the latter are not linearly separable from each other. Four attributes are in this database: sepal length, sepal width, petal length and petal width (all of these measured in cm).

**Functions.** Two classes are defined in this problem with the help of algebraic and trigonometric functions. A total of 88 points with 5 attributes was defined for each class, where these values were randomly generated. The range for each point was $[0,\pi]$ . The functions sin(.), cos(.), tan(.), ln(.) and sqrt(.) were applied to attributes 1, 2, 3, 4 and 5, respectively, for each instance in the case of class 1. Likewise, the functions senh(.), cosh(.), tanh(.), exp(.) and sqr(.) were applied to each object of class 2. Classes 1 and 2 were defined as the sum of their respective functions and the outputs for class 1 and 2 were set to 1 and –1, respectively. The number of objects in the sample was 176: 150 for training and 26 for testing. We believe the contribution of these functions is to prove the accuracy of this method in functions that have not any particular pattern, since the values for the selected attributes were randomly generated.

## 5 Experiments and Results

In the column "problem" of Tables 1, 2, 3 and 4 the codes i_j correspond to the results of one-versus-one classifier for i=1,2 and j=2, 3 (i=1 and j=2 in the case of Table 4). In the case of one-versus-all classifier, i=1,2,3 and j=A ("All"). For instance, 1_2 means "class 1 vs. class 2"; 3_A means "class 3 vs. all", etc.

### 5.1 Lung Cancer

Because 3 classes are considered in this problem, one-versus-one classifier is used in order to test the methodology proposed here. The result of the application of this classifier is shown in Table 1. Results were: 91.2% of average accuracy for training data and 88.9% for test data where splines were applied and 92.3% of average accuracy when the natural spline interpolation was not applied.

### 5.2 Wine Recognition

One-versus-one and one-versus-all classifiers were used in this problem. As mentioned above, 3 classes are considered in this problem. Hence, this results in 3 SVM classifiers for each alternative. The comparison between them is shown in Table 2. The accuracy of both classifiers is good, but the one-versus-one classifier has a better accuracy with an average of 93.6% for training data and 94.3% for test data. For one-

[3] Idem

versus-all classifier the average accuracy for training data is 80.2% and 84.6% for test data.

**Table 1.** Results for Lung Cancer Classification Problem

| | Problem | Sample | | | C | Rest 0 | Training Accuracy | Test Accuracy | Kernel |
|---|---|---|---|---|---|---|---|---|---|
| | | Total | Train | Test | | | | | |
| Original | 1_2 | 27 | 27 | | 16.005 | 0.181 | 92.6% | | RBF 2 |
| | 2_3 | 19 | 19 | | 4.063 | 0.460 | 84.2% | | RBF 2 |
| | 1_3 | 17 | 17 | | 16.000 | -0.160 | 100.0% | | RBF 2 |
| Splines | 1_2s | 100 | 85 | 15 | 128.000 | -0.010 | 84.7% | 93.3% | RBF 2 |
| | 2_3s | 70 | 60 | 10 | 144.000 | 0.004 | 90.0% | 80.0% | RBF 2 |
| | 1_3s | 100 | 85 | 15 | 192.001 | -0.004 | 98.8% | 93.3% | RBF 2 |

**Table 2.** Results for Wine Recognition Problem

| Problem | Sample | | | C | Rest 0 | Training Accuracy | Test Accuracy | Kernel |
|---|---|---|---|---|---|---|---|---|
| | Total | Train | Test | | | | | |
| 1_2 | 130 | 111 | 19 | 160.016 | -0.007 | 95.5% | 94.7% | RBF 2 |
| 1_3 | 107 | 91 | 16 | 192.500 | -0.004 | 100.0% | 100.0% | RBF 2 |
| 2_3 | 119 | 102 | 17 | 192.000 | -0.009 | 85.3% | 88.2% | RBF 2 |
| 1_A | 176 | 150 | 26 | 160.000 | -0.001 | 89.3% | 88.5% | RBF 2 |
| 2_A | 176 | 150 | 26 | 128.000 | -0.010 | 58.0% | 69.2% | RBF 2 |
| 3_A | 176 | 150 | 26 | 224.000 | 0.005 | 93.3% | 96.2% | RBF 2 |

### 5.3 Iris Plant

One-versus-one and one-versus-all classifiers were used for this recognition problem. The results for this classification problem are shown in Table 3. Because one of the classes is linearly separable, one-versus-one classifier offers a better accuracy than one-versus-all classifier. The reason is that the linearly separable class (iris setosa) shows a 100% accuracy when compared with each of the other classes. The average accuracy for training set was 96.1% and for testing set was 93.3% in the case of one-versus-one. In the case of one-versus-all, 90.9% for both training sample and testing sample.

### 5.4 Functions

This is a binary classification problem. The results are shown in Table 4. The accuracy for training set was 97.7% and 100% for testing set.

**Table 3.** Results of GSVM for Iris Plant Classification Problem.

| Problem | Sample | | | C | Rest 0 | Training Accuracy | Test Accuracy | Kernel |
|---|---|---|---|---|---|---|---|---|
| | Total | Train | Test | | | | | |
| 2_3 | 100 | 85 | 15 | 176.000 | -0.001 | 88.2% | 80.0% | Poly 2 |
| 1_3 | 100 | 85 | 15 | 192.000 | 0.000 | 100.0% | 100.0% | RBF 2 |
| 1_2 | 100 | 85 | 15 | 64.000 | -0.006 | 100.0% | 100.0% | RBF 2 |
| 1_A | 150 | 128 | 22 | 128.000 | 0.000 | 100.0% | 100.0% | RBF 2 |
| 2_A | 150 | 128 | 22 | 172.125 | 0.005 | 86.7% | 86.4% | Poly 2 |
| 3_A | 150 | 128 | 22 | 128.000 | 0.000 | 85.9% | 86.4% | Poly 2 |

**Table 4.** Results for the Functions problem.

| Problem | Sample | | | C | Rest 0 | Training Accuracy | Test Accuracy | Kernel |
|---|---|---|---|---|---|---|---|---|
| | Total | Train | Test | | | | | |
| 1_2 | 150 | 128 | 22 | 224.000 | 0.001 | 97.7% | 100.0% | RBF 2 |

## 6 Conclusions

A GSVM classifier is presented in this paper. The application of this algorithm to a set of test problems resulted in a very good performance. The application of a VGA allows us to tackle an extremely complex constrained optimization problem (if judged from the traditional point of view) in a very simple and straightforward way. Consider that every one of the data vectors determines a constraint. For example, in a typical problem the number of constraints is larger than 150. VGA has to determine the band of feasible values out of a potentially infinite set. However, the most important issue is that the value of the regularization parameter was quasi-optimally found trough the algorithm rather than by hand. The reported work seems to indicate that VGA (along with proper constraint handling) is an efficient way to optimize C by including it in the genome. In the past, the difficulty of properly determining the value of C was usually interpreted, simply put, as changing one typical problem in NNs (the determination of the most adequate architecture) into another, perhaps more difficult, one (the best determination of the regularization parameter). If C's determination may be automated as we have shown, then the theoretical advantages of SVMs may be fully exploited and the negative criticism mentioned above may be eliminated.

## Acknowledgement

We want to thank UCI-MLR for the use of Lung Cancer Database, Wine Recognition Database and Iris Plant Database.

## References


1. Jordaan, E. M. & Smits, G. F.: Estimation of the regularization parameter for support vector regression. Proc. of World Conference on Computational Intelligence I. Honolulu, Hawaii. (2002) 2785-2791.
2. Schmitdt, M., Grish, H.: Speaker identification via support vector classifiers. Proc. of International Conference on Acoustics, Speech and Signal Processing, (1996) 105-108.
3. Drucker, H., Wu, D., Vapnik, V.: Support vector machine for spam categorization. Trans. on Neural Networks, Vol. 10. IEEE, (1999) 1048-1054.
4. Vapnik, V., Golowich, S., Smola A.: Support vector method for function approximation, regression, estimation and signal processing.Adv.Neural Inform.Process.Syst., Vol.9. (1996) 281-287
5. Haykin, S.: Neural Networks. A comprehensive foundation. 2nd ed. Prentice Hall, New Jersey (1999)
6. Mercer, J.: Functions of positive and negative type, and their connection with the theory of integral equations. Transactions of the London Philosophical Society (A), Vol.209.(1909) 415-446.
7. Shawe-Taylor, J., Cristianini, N,: Kernel Methods for Pattern Analysis. Cambridge University Press, UK (2004).
8. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, UK (2000).
9. Burges, C. J. C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2, Vol. 2. (1998) 121-167.
10. Cristianini, N., Shawe-Taylor, J., op. cit. pp. 79-92.
11. Cristianini, N., Shawe-Taylor, J., op. cit. pp. 93-124.
12. Haykin, S., op. cit., pp. 318-350.
13. Haykin, S., op. cit., pp. 326-329.
14. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge, Massachusetts (1996).
15. Burges, C. J. C.: op. cit. 121-167.
16. Cristianini, N., op. cit., pp. 93-124.
17. Kuri, A., Gutiérrez, J.: Penalty Functions Methods for Constrained Optimisation with Genetic Algorithms.A Statistical Analysis.Lecture Notes in Artificial Intelligence, Vol.2313. Springer-Verlag, (2002) 108-117.
18. Kuri, A., Mejía, I.: Determination of the Regularization Parameter for Support Vector Machines via Vasconcelos' Genetic Algorithm. Transactions on Circuits and Systems, Issue 4, Vol. 4, WSEAS, (2005) 281-286.
19. Kuri, A., Gutiérrez, J., op. cit., pp.109-111.
20. Fung, G., Mangasarian, O. L.: Multicategory proximal support vector machine classifiers. Data Mining Institute Technical Report, (2001) 01-06.
21. Allwein, E. L., Shapire, R. E., Singer, Y.: Reducing multiclass to binary: A unifying approach for margin classifiers.Journal of Machine Learning Research, Vol.1, (2000) 113-141.
22. Bojanov, B., Hakopian, H., Sahakian, B.: Spline Functions and Multivariate Interpolations. Springer-Verlag, (1993).

# Natural Language Processing

# Language Portable Detection for Spanish Named Entities

Zornitsa Kozareva, Oscar Ferrández, Andrés Montoyo and Rafael Muñoz

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante, Spain
{zkozareva,ofe,montoyo,rafael}@dlsi.ua.es

**Abstract.** We propose a language portable Named Entity detection module developed and tested over Spanish and Portuguese. The influence of different feature sets over the classification task was studied and demonstrated. The differences in language models learned by three data-driven systems performing the same NLP tasks were examined. They were combined in order to yield a higher accuracy than the best individual system. Three NE classifiers (Hidden Markov Models, Maximum Entropy and Memory-based learner) are trained on the same corpus data and after comparison their outputs are combined using voting strategy. Results are encouraging since 92.96% f-score for Spanish and 78.86% f-score for Portuguese language portable detection were achieved. For Spanish the classification which is based on the language portable detection reached 78.59% f-score. Compared with the systems competing in CoNLL-2002 our system reaches third place.

## 1 Introduction

The increasing flow of digital information requires the extraction, filtering and classification of pertinent information from large volumes of texts. Information Extraction, Information Retrieval and Question Answering systems need Named Entity (NE) recognition and classification modules. For English the available resources and the developed systems outnumber, but in the case of languages as Spanish, Portuguese or eastern European ones where the resources as gazetteers[1], annotated corpora etc. are not sufficient but the need is still the same, the situation looks different. This fact motivated us to start the development of a language resource independent system during its NE detection and using less resources while classifying into LOC, PER or ORG classes.

In this paper we present a NE system developed for Spanish. Three machine learning algorithms were used in concrete: Hidden Markov Model, Maximum Entropy and the Memory-based learner. They were applied to the CoNLL-2002 shared task for Spanish. The language portable detection was also tested with Portuguese language. Both languages come from the Romance language group and have similar behavior so features valid for Spanish were directly adopted by Portuguese.

[1] catalogues of names of people, locations, organizations etc.

© A. Gelbukh, R. Monroy. (Eds.)
Advances in Artificial Intelligence Theory
Research on Computing Science 16, 2005, pp. 219-228

In order to improve overall performance feature selection and systems' combination were done. Aiming at minimal feature space, less processing time and gaining results while restraining from gazetteers, the obtained results are quite encouraging. For Spanish we reached 92.96% f-score for language portable detection and 78.59% f-score for classification. Portuguese was used to support our hypothesis for language portable detection and we gained 78.86%. A study of the occurred errors and proposals for resolving them was made, comparison with existing systems and future work are discussed. The paper is organized as following: in Section 2 we expose the features on which the classification methods are based and a brief description of the classifiers, the voting strategy and the data with which we worked is in Section 3, discussion of the obtained results and error correction during NE detection is demonstrated in Section 4, classification into classes is represented in Section 5, a comparison with CoNLL-2002 systems is exposed in Section 6, we conclude and mention about the future work in Section 7.

## 2 Feature description and Classification methods

For NE detection and classification task, the Memory-based learning and Maximum Entropy classifiers utilize the features described below, HMM takes only the three most informative attributes.

### 2.1 Features for NE detection

We use the well-known BIO model for NE detection, where a tag shows that a word is at the beginning of a NE (B), inside a NE (I) or outside a NE (O). For the sentence: *Paulo Suarez es mi amigo.* , the following tags have been associated, "B I O O O O ", where *Paulo* starts a named entity; *Suarez* continues this entity; while the words *es*, *mi*, *amigo* and the full stop are not part of a NE.

The original set for BIO is composed of 29 features as described in Figure 1 and we denote this set by $A$. For improving classifier's performance different feature combinations of the original set were constructed. The features represent words, position in a sentence, capitalization, suffixes, context information, lists of entity triggers for NE. The features c[1-6], C[1-7], d[1-3] refer to the words in a $\{-3, +3\}$, window of the anchor word **a**.

We extracted two, three and half substrings of the anchor word, knowing that some prefixes and suffixes are good indicators for certain classes of entities, taking into account the morphological structure of a word and its paradigm. In general suffixes are more informative, for Spanish endings as *-er,-or,-ista* imply person's occupation *pianista, futbolista, profesor, director* and can help during detection and classification phase. It is surprising the number of Spanish surnames that end in *-ez*, meaning "son of", like the suffix *-son* and *-sen* in many German and Scandinavian languages, *-ov,-ova,-ev,-eva* in Russian and Bulgarian, and-*es* in Portuguese. (e.g. Fernandez is the son of Fernando [Ferdinan]). Of course these examples have many exceptions, but the information they contribute is

- **a**: anchor word (e.g. the word to be classified)
- ***c[1-6]***: word context at position ±1, ±2, ±3
- ***C[1-7]***: word capitalization at position 0, ±1, ±2, ±3
- ***d[1-3]***: word +1,+2,+3 in dictionary of entities
- ***p***: position of anchor word
- ***aC***: capitalization of the whole anchor word
- ***aD***: anchor word in any dictionary
- ***aT***: anchor word in dictionary of trigger words
- ***wT***: word at position ±1, ±2, ±3 in a dictionary of trigger words
- ***aL***: lema of the anchor word
- ***aS***: stem of the anchor word
- ***aSubStr[1-5]***: ±2, ±3 and half substring of the anchor word

**Fig. 1.** Features for NE detection

significant when combined with other features. The lemma expands the search in the gazetteers' list we maintain, we can have the word "profesor" but not "profesora" and by the lemma which returns the base of the word, we are going to have a positive vote.

### 2.2 Features for NE classification

The tags used for NE classification are PER, LOC, ORG and MISC as defined by CoNLL-2002 shared task. For classification, the first seven features used by the BIO model (e.g. a, c[1-6], p) were used as well as the additional set described in Figure 2. The gazetteers for the attributes gP, gL and gO have been collected randomly from cites as yellow pages.

- ***eP***: entity is trigger PER
- ***eL***: entity is trigger LOC
- ***eO***: entity is trigger ORG
- ***eM***: entity is trigger MISC
- ***tP***: word ±1 is trigger PER
- ***tL***: word ±1 is trigger LOC
- ***tO***: word ±1 is trigger ORG
- ***gP***: part of NE in gazetteer for PER
- ***gL***: part of NE in gazetteer for LOC
- ***gO***: part of NE in gazetteer for ORG
- ***wP***: whole entity is PER
- ***wL***: whole entity is LOC
- ***wO***: whole entity is ORG
- ***NoE***: whole entity not in one of the defined three classes
- ***f***: first word of the entity
- ***s***: second word of the entity
- ***clx***: capitalization, lowercase, other symbol

**Fig. 2.** Features for NE classification

### 2.3 Classification methods

For NE detection we worked with Memory-based learning and Hidden Markov Model, while for NE classification we had also Maximum Entropy.

The memory-based software package we used is called TiMBL [3]. Its default learning algorithm, instance-based learning with information gain weighting (IB1IG) was applied. The Hidden Markov Models toolkit ICOPOST[2] developed by [8] has been functioning for POS tagging, but we adapted it for NER. The maximum entropy classifier we worked with was a very basic one with no smoothing or feature selection, implemented in C++ by [9].

## 3 Classifier combination and Data

### 3.1 Classifier combination

It is a well-known fact that if several classifiers are available, they can be combined in various ways to create a system that outperforms the best individual classifier. Since we had several classifiers available, it was reasonable to investigate combining them in different ways. The simplest approach to combining classifiers is through voting, which examines the outputs of the various models and selects the classifications which have a weight exceeding some threshold, where the weight is dependent upon the models that proposed this particular classification. It is possible to assign varying weights to the models, in effect giving one model more importance than the others. In our system, we assigned to each model the weight corresponding to the correct class it determines.

### 3.2 Data and its evaluation

The Spanish train and test data we used are part of the CoNLL-2002 [7] corpus. For training we had corpus containing 264715 tokens and 18794 entities and for testing we used Test-B corpus with 51533 tokens and 3558 entities.

The Portuguese corpus we used is part of HAREM[3] competition with 68597 tokens and 3094 entities for training, and 22624 tokens and 1013 entities for testing.

Scores were computed per NE class and the measures used were Precision (of the tags allocated by the system, how many were right), Recall (of the tags the system should have found, how many did it spot) and $F_{\beta=1}$(a combination of recall and precision). *Conlleval* evaluation script was used in order to have comparable results with the CoNLL-2002 systems.

[2] http://acopost.sourceforge.net/
[3] http://poloxldb.linguateca.pt/harem.php

## 4 NE recognition by BIO model

Our NER system is composed of two passages

1. detection: identification of sequence of words that makes up the name of an entity.

2. classification: deciding to which category our previously recognized entity should belong.

For NE detection we follow the BIO model described briefly in subsection 2.1. Our experiments with TiMBL started using set $C24 = A/\{aSubStr[1-5]\}$, which contained all attributes as lemma, dictionaries, trigger words etc. The obtained results have been satisfactory as can be seen in Table 1, but since we have been searching for an appropriate feature set $F$ that maximizes the performance, minimizes the computational cost and being language portable, we made a study of the features and selected the most informative ones according to the information gain measure. Four candidate sets were formed and we denote them by $C24r = \{a, c[1-6], C[1-7], p, aC, wD, wT, aL, aS\}$ and $C17 = C24r/\{c[5-6], C[6-7]\}$; considered as language dependent (they use dictionaries, tools as lemmatizers, stemmers etc.) and $E12 = \{a, c[1-4], C[1-5], p, aC\}$ and $E17 = E12 \cup \{aSubStr[1-5]\}$, considered as language portable. The results of each individual set can be seen in Table 1.

| Tags | B(%) | | | I(%) | | | BIO(%) | | |
|---|---|---|---|---|---|---|---|---|---|
| Classifier | Prec. | Rec. | $F_{\beta=1}$ | Prec. | Rec. | $F_{\beta=1}$ | Prec. | Rec. | $F_{\beta=1}$ |
| TMB-C24 | 94.42 | 95.19 | 94.81 | 87.25 | 85.67 | 86.45 | 92.51 | 92.61 | 92.56 |
| TMB-C17 | 94.47 | 95.11 | 94.79 | 87.28 | 85.37 | 86.31 | 92.56 | 92.47 | 92.51 |
| TMB-C24r | 94.63 | 94.01 | 94.32 | 87.99 | 85.07 | 86.50 | 92.86 | 91.58 | 92.22 |
| HMM-CD | 92.18 | 93.82 | 92.99 | 83.94 | 81.98 | 82.95 | 90.01 | 90.60 | 90.31 |
| HMM-CW | 92.40 | 93.99 | 93.19 | 83.71 | 81.00 | 82.33 | 90.13 | 90.46 | 90.29 |
| Vote 1 ld | 95.31 | 95.36 | 95.34 | 88.02 | 87.56 | 87.79 | 93.34 | 93.24 | 93.29 |
| TMB-E12 | 94.33 | 94.91 | 94.62 | 87.00 | 85.29 | 86.14 | 92.38 | 92.30 | 92.34 |
| TMB-E17 | 94.17 | 95.28 | 94.72 | 87.62 | 85.37 | 86.48 | 92.44 | 92.59 | 92.51 |
| HMM-CW | 92.40 | 93.99 | 93.19 | 83.71 | 81.00 | 82.33 | 90.13 | 90.46 | 90.29 |
| Vote 2 li | 94.43 | 95.73 | 95.07 | 88.31 | 86.05 | 87.17 | 92.81 | 93.10 | 92.96 |

**Table 1.** *BIO for Spanish*

Initially to HMM we passed the NE and the tag associated with it. The obtained performance of 88.63% is less than each one of TiMBL's individual sets, however this difference is compensated with the number of features TiMBL uses. For the word *Don Simon*, which in one text can mean a name of a person or organization (e.g. company name), in order to determine its correct significance more information is needed. One advantage of HMM is its time performance of several minutes in comparison with the other methods, but fails in adding lots of features. As studied by Rössler [6] to HMM features can be passed by

| Tags | B(%) | | | I(%) | | | BIO(%) | | |
|---|---|---|---|---|---|---|---|---|---|
| Classifier | Prec. | Rec. | $F_{\beta=1}$ | Prec. | Rec. | $F_{\beta=1}$ | Prec. | Rec. | $F_{\beta=1}$ |
| TMB-E12 | 82.50 | 83.32 | 82.91 | 72.77 | 64.77 | 68.53 | 79.59 | 77.26 | 78.41 |
| TMB-E17 | 80.13 | 83.22 | 81.65 | 69.64 | 58.86 | 63.80 | 77.16 | 75.27 | 76.20 |
| HMM-CW | 77.83 | 68.61 | 72.93 | 61.02 | 58.66 | 59.81 | 72.01 | 65.36 | 68.53 |
| Vote 3 li | 82.35 | 84.30 | 83.32 | 72.75 | 65.78 | 69.09 | 79.47 | 78.26 | 78.86 |

**Table 2.** *Language portable BIO for Portuguese*

corpus or tag transformation. We studied both possibilities and saw that tag transformation gave better results. We took the two most informative attributes - word capitalization and whole word in capitals, plus the gazetteer list and passed them as features to the B and I tags. For *La Coruña* we have B-XX and I-XX tags, where the XX take binary features. With HMM-CD we denote the results after passing the attributes word capitalization and word in dictionary and with HMM-CW the results from word capitalization and whole word in capitals. Adding these attributes, HMM's performance increases with around 1.68%.

The obtained results from all BIO sets for Spanish can be observed in Table 1, there we mention the language dependent sets for comparison, but for further experiments (classification) we consider the results from the language portable sets. In Table 2 we demonstrate Portuguese language portable BIO detection using the same sets as for Spanish.

The coverage of tag O is high due to its frequent appearance, however its importance is not so significant as the one of B and I tags, who actually detect the entities. For this reason we demonstrate separately system's precision, recall and f-score for B and I tags. The best score for Spanish BIO was obtained by TiMBL considering the complete $C24$ set with f-score of 92.56%. Comparing this score with set $C17$ where he number of features is reduced, the word window diminished from $\pm 3$ to $\pm 2$, the difference of 0.05% is insignificant. Set $C24r$ was studied for reducing some noisy attributes from set $C24$ but still keeping the $\pm 3$ window. Its total BIO performance decreased but gained 86.50% - the highest f-score per I tag.

The language portable sets perform quite similar to the dependent ones. For tag B, set E12 with its 12 attributes performs better than C24r. The complete BIO for E12 is better than those of C24r. TMB-E17 improves slightly the overall results of E12 and has similar results to C17. For tag I it performs better than C24, C17 and has 0.02% less performance than C24r.

The classifiers used different feature sets and we noticed that one classifier detects an entity while the other doesn't. The classifiers used different feature sets and we noticed that one classifier detects an entity while the other doesn't. After obtaining the different results we applied voting techniques grouping the language dependent sets in vote one and the language portable sets in vote two. The difference of 0.33% between *Vote 1 language dependent* with 93.29% performance and *Vote 2 language portable* with 92.96% f-score shows how small

| Tags | LOC(%) | | | MISC(%) | | | ORG(%) | | | PER(%) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Classifier | Prec. | Rec. | $F_{\beta=1}$ | Prec. | Rec. | $F_{\beta=1}$ | Prec. | Rec. | $F_{\beta=1}$ | Prec. | Rec. | $F_{\beta=1}$ |
| ME-F24 | 81.16 | 74.72 | 77.81 | 69.29 | 49.12 | 57.49 | 74.21 | 84.07 | 78.83 | 82.95 | 88.03 | 85.41 |
| TMB-F24 | 75.70 | 75.28 | 75.49 | 55.03 | 51.47 | 53.19 | 75.22 | 79.79 | 77.44 | 84.53 | 83.27 | 83.89 |
| ME-F24clx | 81.94 | 74.91 | 78.27 | 69.67 | 50.00 | 58.22 | 73.92 | 84.00 | 78.64 | 83.18 | 88.16 | 85.60 |
| TMB-F24clx | 74.84 | 75.46 | 75.15 | 55.88 | 50.29 | 52.94 | 75.88 | 79.79 | 77.79 | 85.42 | 85.31 | 85.36 |
| TMB-R24 | 80.08 | 75.65 | 77.80 | 57.95 | 48.24 | 52.65 | 77.01 | 81.36 | 79.12 | 79.24 | 88.30 | 83.53 |
| TMB-R24clx | 79.20 | 75.18 | 77.14 | 63.20 | 50.00 | 55.83 | 76.14 | 81.36 | 78.66 | 80.15 | 88.44 | 84.09 |
| HMM | 74.85 | 67.80 | 71.15 | 44.66 | 46.76 | 45.69 | 72.06 | 73.86 | 72.95 | 66.11 | 74.83 | 70.20 |
| VM24T24fclxH | 81.16 | 75.92 | 78.46 | 66.80 | 49.71 | 57.00 | 75.06 | 83.21 | 78.93 | 83.72 | 89.52 | 86.52 |

**Table 3.** *NE classification*

feature set containing attributes independent from any tools, dictionaries or gazetteers can give good and similar results to the dependent sets.

Taking in mind that Spanish and Portuguese are languages having similar behavior, we studied and saw how attributes valid for Spanish were directly adopted by Portuguese. In Table 2 we to show the results for Portuguese after applying the same set of portable features as for Spanish. With voting 83.32% f-score for B tag and 78.86% for complete BIO were achieved. These results are acceptable since we didn't have sufficient training data and the annotated corpus we used had significant number of errors.

### 4.1 BIO error Analysis

After analyzing the obtained results, we saw that some of the occurred errors can be avoided by applying simple post-processing: when an I tag has been preceded by O tag we substituted it by B if the analyzed word starts with a capital letter and in the other case we simply put O; sequences such as $OBIBI$, have been transformed into $OBIII$. (see the example in subsection 2.1).

For Spanish around 2% of the errors came from the annotated corpus, sometimes quotation mark symbol was annotated as B and sometimes as O. Portuguese corpus was quite noisy having entity as *v+,n 12* annotated as organization or some names of people were even not annotated.

One of our attributes concerns word capitalization and had great impact over the detection task. We noticed how sometimes words starting a sentence but not belonging to any of the named entity classes were classified as B tags. A statistical study of word frequency, determines if a word at the beginning of a sentence should have a B tag or not. The word variants (e.g. writing of a word), their individual frequency and neighbors with which these words appear are studied. Thus we have been able to correct and avoid this kind of error.

| Tags | LOC(%) | | | MISC(%) | | | ORG(%) | | | PER(%) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Classifier | Prec. | Rec. | $F_{\beta=1}$ | Prec. | Rec. | $F_{\beta=1}$ | Prec. | Rec. | $F_{\beta=1}$ | Prec. | Rec. | $F_{\beta=1}$ |
| ourNE | 81.16 | 75.92 | 78.46 | 66.80 | 49.71 | 57.00 | 75.06 | 83.21 | 78.93 | 83.72 | 89.52 | 86.52 |
| WNC | 79.15 | 77.40 | 78.26 | 55.76 | 44.12 | 49.26 | 74.73 | 79.21 | 76.91 | 80.20 | 89.25 | 84.48 |
| CY | 79.66 | 73.34 | 76.37 | 64.22 | 38.53 | 48.16 | 76.79 | 81.07 | 78.87 | 82.57 | 88.30 | 85.34 |
| Flo | 82.06 | 79.34 | 80.68 | 59.71 | 61.47 | 60.58 | 78.51 | 78.29 | 78.40 | 82.94 | 89.93 | 86.29 |
| CMP | 85.76 | 79.43 | 82.43 | 60.19 | 57.35 | 58.73 | 81.21 | 82.43 | 81.81 | 84.71 | 93.47 | 88.87 |

**Table 4.** *CoNLL-2002 NE classification*

| Classifier | Prec. % | Rec. % | $F_{\beta=1}$ % |
|---|---|---|---|
| CMP | 81.36 | 81.40 | 81.39 |
| Flo | 78.70 | 79.40 | 79.05 |
| ourNE | 78.09 | 79.10 | 78.59 |
| CY | 78.19 | 76.14 | 77.15 |

**Table 5.** *Complete system performance*

## 5 NE classification

After detection follows NE classification into LOC, MISC, ORG or PER class as defined by CoNLL-2002. For this task, we used the results obtained from the language portable detection.

For ME and TiMBL, we started the classification with a set composed of 24 features as described in subsection 2.2. Let us denote by $F24$ the set having features: a, c[1-6], p, eP, eL, eO, eM, tP, tL, tO, gP, gL, gO, wP, wL, wO, NoE, f and s. In Table 3 comparing the performance of ME and TiMBL with the same set can be seen how ME classifies better for each one of the classes.

Choosing the most informative attributes,{a, c[1], eP, gP, gL, gO, wP, wL, wO, NoE, f}, we create a set $R24$, where $R24 \subset F24$. In Table 3 we displayed only the results obtained by TiMBL, because ME needs a lot of time for training and testing. When both classifiers were compared on small random samples from the original set, we saw that TiMBL performs better with the reduced set. When $R24$ was tested with the complete data TiMBL achieved the highest result for ORG class of 79.12%. Two additional sets $R24clx = R24 \cup \{clx\}$ and $F24clx = F24 \cup \{clx\}$, where $clx$ is the attribute described in Figure 2, were constructed. $R24clx$ lowered the performance for LOC and ORG class compared to the $R24$ set but performed better dealing with MISC and PER class. By adding $clx$ attribute to $F24$, ME improved its performance with 0.46% for LOC and 0.19% for PER class and gained the maximum score of 58.22% for MISC class. Using the same set TiMBL decreased its score for LOC and MISC class and slightly improved ORG and PER classes. Among all classifiers, HMM has the lowest score per class. The voting we applied considers ME-F24, Timbl-F24clx and HMM results.

We have seen how the elimination or addition of features gave impact over given types of classes during classification. As a whole our systems perform well

when classifying into PER,ORG and LOC class, but not when dealing with MISC class which is difficult to be detected due to its heterogeneity.

## 6 Comparison with CoNLL-2002 systems

We demonstrated the performance of NER considering different machine learning methods, where the advantages and disadvantages of each one of them being in time performance or feature maintenance was shown. Apart from this it is very interesting to make a comparative study with the systems participating in CoNLL-2002 shared task, since our system has been developed using the same data; we should take in mind that our classification has been based on language portable detection.

Table 4 represents the results per class for our system and the first four best performing systems in CoNLL-2002; WNC[4], CY[2], Flo[5], CMP[1]. When classifying into LOC class our system performed with 0.2% and 2.09% better than the one of Wu and Cucerzan and less with 2.22% and 3.97% from the systems of Florian and Carreras. Our classification into MISC class was better with 7.74% and 8.84% compared to the one of Wu and Cucerzan and less with 3.58% and 1.73% from Florian and Carreras. For ORG and PER classes we outperformed all systems except the one of Carreras. With Wu's system we have 2.02% and 2.04% better score per ORG and PER class, from Cucerzan's 0.06% and 1.18% and from the system of Florian 0.53% and 0.23%.

We separated the overall performance of the first three best performing systems in Table 5. Comparing the f-score our system performs with 1.44% better than the third one, with 0.46% less than the second and with 2.8% less than the first system.

## 7 Conclusions and future work

We presented a combination of machine learning methods (Memory-based learning, Maximum Entropy and HMM) for performing NE detection and classification task for Spanish. Aiming at minimal feature space and restraining from dictionaries or other language dependent tools, we demonstrated one language portable detection for Spanish and Portuguese. The Portuguese system served as proof for our experiments and hypothesis. At present we didn't study the achievement of language portable classification over Spanish and we depend on gazetteers but in future we intend to work on this task. Comparing our results with CoNLL-2002 participants the f-score results of 78.46% for LOC, 57.00% for MISC, 78.93% for ORG and 86.52% for PER are quite encouraging and are among the second and third system.

As future work we intend to develop and use specific dictionaries for NEs, to apply the same method for languages as Catalan, Italian and French. We are interested in dividing the original three base tags into more detailed ones, for example: ORG class into administration, institution, company etc. A Word Sense Disambiguation module is going to be included and the rule based system that

was separately developed and deals with weak entities such as *El presidente del Gobierno de La Rioja* is going to be merged with the machine learning module we have developed.

## Acknowledgements

This research has been funded by the Spanish Government under project CICyT number TIC2003-07158-C04-01 and PROFIT number FIT-340100-2004-14 and by the Valencia Government under project numbers GV04B-276 and GV04B-268.

## References

1. Xavier Carreras, Lluís Màrques, and Lluís Padró. Named entity extraction using adaboost. In *Proceedings of CoNLL-2002*, pages 167–170. Taipei, Taiwan, 2002.
2. Silviu Cucerzan and David Yarowsky. Language independent ner using a unified model of internal and contextual evidence. In *Proceedings of CoNLL-2002*, pages 171–174. Taipei, Taiwan, 2002.
3. Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. TiMBL: Tilburg Memory-Based Learner. Technical Report ILK 03-10, Tilburg University, November 2003.
4. Marine Carpuat Jeppe Larsen Dekai Wu, Grace Ngai and Yongsheng Yang. Boosting for named entity recognition. In *Proceedings of CoNLL-2002*, pages 195–198. Taipei, Taiwan, 2002.
5. Radu Florian. Named entity recognition as a house of cards: Classifier stacking. In *Proceedings of CoNLL-2002*, pages 175–178. Taipei, Taiwan, 2002.
6. M. Rössler. Using markov models for named entity recognition in german newspapers. In *Proceedings of the Workshop on Machine Learning Aproaches in Computational Linguistics*, pages 29–37. Trento, Italy, 2002.
7. Tijong Kim Sang. Introduction to the conll-2002 shared task: Language independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158, 2002.
8. Ingo Schröder. A case study in part-of-speech tagging using the icopost toolkit. Technical Report FBI-HH-M-314/02, Department of Computer Science, University of Hamburg, 2002.
9. Armando Suárez and Manuel Palomar. A maximum entropy-based word sense disambiguation system. In Hsin-Hsi Chen and Chin-Yew Lin, editors, *Proceedings of the 19th International Conference on Computational Linguistics, COLING 2002*, pages 960–966, August 2002.

# Boosting Applied to Spanish Word Sense Disambiguation

Rocio Guillén

California State University San Marcos, San Marcos CA 92096, USA

**Abstract.** Recent trends in word sense disambiguation (WSD) show that the most effective approach is that of machine learning (ML). Ensemble learning methods such as boosting select a collection of hypotheses from the hypothesis space and combine their prediction. Boosting algorithms combine many weak hypotheses to find a highly accurate classification rule. In this paper we describe a system that applies a boosting algorithm to the WSD problem and present results from the SENSEVAL-3 exercise for the Spanish lexical-sample task. Our system SenseFinder utilizes Schapire and Singer's Boostexter [3] implementation of the AdaBoost.MH algorithm as the learning paradigm. We work on a set of 46 polysemous words and use tagged and lemmatized files from which we extract a window of 5 lemmas. This information is used to describe the examples and to train our system.

## 1 Introduction

Word sense disambiguation (WSD) involves the mapping of a given word in a text or discourse to a definition or meaning, i.e., sense, which is distinguishable from other meanings potentially assignable to that word. The task is twofold: determining all the different senses for every word, and finding a method to assign each word to the appropriate sense [2]. Sense disambiguation is essential for natural language processing tasks such as text processing, speech processing, human-computer interaction, message understanding, information retrieval and machine translation.

A wide range of symbolic and statistical or machine learning methods have been explored to solve the problem of WSD. Despite the research efforts devoted to tackle the problem there is no large-scale, broad coverage and highly accurate word sense disambiguation system available at present.

In this paper, we present the use of a machine learning approach, called boosting, to the problem of WSD. Boosting is a general method for improving the accuracy of learning algorithms. It is based on the observation that finding and combining many simple and moderately accurate "rules of thumb" is easier than finding a single, highly accurate prediction rule. [5] To find these rules, a weak or base learning method or algorithm is applied. The boosting algorithm calls the weak learner repeatedly, each time with a different distribution or weighting over the training examples. At each iteration, the weak learner generates a new weak prediction rule, and after many iterations the boosting algorithm combines

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 229-238*

the weak rules into a single prediction rule. The distribution at each iteration is chosen by placing the most weight on the examples most often misclassified by the preceding weak rules. The combination of weak rules is done by taking a weighted majority vote.

AdaBoost, a boosting algorithm introduced by Freund and Schapire [4], has been studied extensively and has been shown to perform well [7],[4],[8],[6],[5] on standard machine-learning tasks using standard machine-learning algorithms as weak learners. The first versions of the algorithm, AdaBoost.M1 and Adaboost.M2, only supported output data belonging to one class at most. Extensions to the algorithm, AdaBoost.MH and AdaBoost.MR, were designed to solve the problem of an example belonging to various classes. The goal of AdaBoost.MH is to predict all and only the correct labels, senses in our work. The goal of AdaBoost.MR is to find a hypothesis which ranks labels so that it hopefully places the correct labels (senses) at the top of the ranking. BoosTexter is a system which implements four versions of boosting based on these extensions that we have used to test our system.

AdaBoost.MH has been successfully applied to natural language processing problems such as word sense disambiguation [10], human-computer spoken-dialogue systems [11],[12] and part-of-speech tagging and prepositional phrase attachment disambiguation [13] as well as information retrieval tasks like text categorization [3] and document routing [14]. Additionally, AdaBoost has been proven to be theoretically well founded.

The paper is organized as follows. We first present in Section 2 research done in WSD using Boosting algorithms. In Section 3 we briefly describe the AdaBoost.MH algorithm and the BoosTexter system. In Section 4 we describe SenseFinder, the domain of application and evaluation metric. Experiments using the BoosTexter system are described, and results and evaluation of results are presented in Section 5. Lastly, Section 6 presents conclusions and future work.

## 2 Boosting algorithms for WSD

Supervised learning has become the most successful approach to tackle the problem of WSD. These algorithms follow a two-step process. The first step is to choose the representation of the context of the target word senses as a set of features. Then apply a ML algorithm to train on the chosen features and assign a sense to the target word in the test examples. Among the supervised learning algorithms that have been applied to WSD are Naive Bayes [10], Exemplar-based [15], Decision Lists [16], and Neural Networks [17].

Supervised learning algorithms suffer from high overhead for supervision and additional overhead for learning/testing when scaling to real size WSD problems. Due to this fact, research on reducing the need for supervision of corpus-based methods for WSD is currently under way. Escudero et al. [10] have worked on reducing the feature space for English. They have developed a variant of AdaBoost.MH, called LazyBoosting, which has been tested on a

medium/large sense-tagged corpus containing about 193K examples of the 191 most frequent and ambiguous English words. Results showed that boosting compares favourably to the Naive Bayes and the Exemplar-based approach.

LazyBoosting is a simple modification of the AdaBoost.MH algorithm, which consists of reducing the feature space that is explored when learning each weak classifier. More specifically, a small subset $S$ of features/attributes are randomly selected and the best weak rule is chosen among them. The idea is that if the subset $S$ is not too small, it is more likely that a sufficiently good rule can be found at each iteration. Additionally, no feature/attribute has to be discarded thus avoiding the risk of eliminating relevant attributes.

The TALP system is based on the LazyBoosting algorithm [10]. The features represent local and topical contents and domain labels. Let $w_i$ be the word to be disambiguated, $\ldots w_{i-3}w_{i-2}w_{i-1}w_iw_{i+1}w_{i+2}w_{i+3}\ldots$ the context of words around $w_i$, and $p_{i\pm j}$, $j = 1,2,3$ be the part-of-speech tag of $w_{i\pm j}$. Local context feature patterns are represented as follows.

$p_{i-3}, p_{i-2}, p_{i-1}, p_{i+1}, p_{i+2}, p_{i+3}, w_{i-2}, w_{i-2}, w_{i-1}, w_{i+1}$,
$w_{i+2}, (w_{i+2}, w_{i-1}), (w_{i-1}, w_{i+1},$ and $(w_{i+1}, w_{i+2})$

The last three patterns represent collocations of two consecutive words.

We apply BoosTexter, an implementation of the AdaBoost.MH algorithm, to train and test the system. We use a subset of features to describe the examples and train the classifiers similar to the one described for the TALP system.

## 3 AdaBoost.MH

In this section we present AdaBoost.MH (see Figure 1) designed by Schapire and Singer's [1],[3],[5],[11], which is an extension to the AdaBoost algorithm. It has been designed to work for multiclass multi-label classification problems.

**Input:** $(x_1, Y_1), \ldots, (x_m, Y_m)$ where $x_i \in \mathcal{X}, \mathcal{Y}) \subseteq \mathcal{Y}$
**Output:** Final hypothesis

$$f(x,l) = \sum_{t=1}^{T} h_t(x,l)$$

**Initialize** $D_1(i,l) = 1/(mk)$
**For** $t = 1, \ldots, T$ **do**

1. Train weak learner using distribution $D_t$ .
2. Weak learner returns weak hypothesis $h_t : \mathcal{X} \times \mathcal{Y} \rightarrow \Re$
3. Update distribution

$$D_{t+1}(i,l) = \frac{D_t(i,l)exp(-Y_i[l]h_t(x_i,l))}{Z_t}$$

where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ will be a distribution)

Figure 1 AdaBoost.MH Algorithm

AdaBoost for single-label classification maintains a set of weights over training examples to force the weak learner to concentrate on examples which are

hardest to classify; whereas for multiclass multi-label classification, the boosting algorithm maintains a set of weights and labels. Training examples and their corresponding labels that are harder to classify get incrementally higher weights. The intention is to force the weak learner to focus on examples and labels that will contribute more to the overall goal of finding a highly accurate classification rule.

Let $S$ be a sequence of training examples $< (x_1, Y_1), \ldots, (x_m, Y_m) >$ where each $x_i$ belongs to some domain or instance space $\mathcal{X}$ and each label $Y_i$ is in some label set $\mathcal{Y}$. Assume that $\mathcal{Y}$= { -1, +1 }. As described above, AdaBoost.MH maintains a set of weights as a distribution $D_t$ over examples and labels. Initially, this distribution is uniform. On each iteration $t$, the distribution is inputted to the weak learner to compute a weak hypothesis $h_t$. The output of the weak learner is a hypothesis $h : \mathcal{X} \times \mathcal{Y} \rightarrow \Re$. Let $l$ be a label, then the sign of $h(x, l)$ is interpreted as a prediction of the value of $Y[l]$, i.e., whether $l$ is or is not assigned to $x$. The magnitude of the prediction denoted $\mid h(x, l) \mid$ is interpreted as a measure of confidence in the prediction. The distribution $D_t$ is updated in a manner that increases the weight of example-label pairs $(x, l)$ which are misclassified by $h_t$. Testing the value of a Boolean predicate and making a prediction based on that value is carried out using very weak hypotheses for WSD following [13] and [10]. The predicates in our system are of the form $f\ v$ where $f$ is a feature (word) and $v$ is its corresponding part-of-speech and lexical features, e.g., *arte NCCS000*. The predicates used are $f_{i-2}\ v_{i-2}$, $f_{i-1}\ v_{i-1}$, $f_i\ v_i$, $f_{i+1}\ v_{i+1}$, $v_{i+2}\ v_{i+2}$. Formally, based on a given predicate $P$, we are interested in weak hypotheses $h$ which make predictions as follows.

$$h(x, l) = \begin{cases} c_{0t} \text{ if } P \text{ holds in } x, & c_{jt} \in \Re \\ c_{1t} \text{ otherwise}, & c_{jt} \in \Re \end{cases}$$

### 3.1 BoosTexter

In practice, the AdaBoost.MH algorithm has been implemented as the BoosTexter system for text categorization tasks. BoosTexter works with data which may be of various forms. In general, each instance or example is split up into multiple fields. These fields may be one of the following four types: continuous-value attribute (e.g., *word-position*), discrete-valued attribute (e.g., *part-of-speech*, text string (e.g., *actuar en Cannes*), or scored-text string (e.g., *word-frequency*).

BoosTexter combines many simple hypotheses (rules) iteratively. Each hypothesis (rule) consists of a simple binary test and predictions for each outcome of the test. Depending on the type of input field the binary test has one of the following forms.

| Type | Test |
|---|---|
| Discrete | Does the attribute have a particular value? |
| Continuous | Is the value of the attribute > threshold or attribute < threshold? |
| Text | Is the string (ngram) in given text? |
| Scored | Is the score of the word > threshold or is the score < threshold? |

The predictions associated with each outcome of a given hypothesis (rule) are described by a set of weights over the possible labels. The weights should not be interpreted as probabilities.

## 4 SenseFinder System

The SenseFinder system was developed for the SENSEVAL-3 Spanish Lexical Sample Task [18]. The purpose of this task is to evaluate supervised and semi-supervised learning algorithms for WSD. Experiments are carried out on a set of 46 words. The examples for all 46 words in both the training and test set have been extracted from the year-2000 corpus of the Spanish EFE News Agency. Each example has been tagged with a unique sense. Additionally, POS tagged and lemmatized files have been provided, in which the contexts of the examples are tokenized, lemmatized and POS tagged.

The system utilizes BoosTexter for describing the examples, training and testing since the WSD problem can be considered as a categorization task in which a word is assigned to a pre-existing set of senses. Our goal is to generate a classifier for each word; each word represents a categorization problem.

The first step before applying the boosting algorithm is to extract the information to describe the examples used for training. Extraction is done by tokenizing the tagged and lemmatized file looking for the word tagged as the head. The next step consists of creating a window that includes the following: withe previous two lemmas and their corresponding part-of-speech; the head's lemma and corresponding part-of-speech; and the following two lemmas and their corresponding part-of-speech. Punctuation marks are ignored and diacritic (accent) marks are removed. A sample input sentence, part of its corresponding tagged sentence, and generated output is shown in Figure 2. Output thus generated become the training examples. The same step is applied to the test set to generate the examples for testing.

Input Sentence:
*Photopainters.com, según los jóvenes empresarios, "no es estrictamente comercial o de arte, sino una web que une cultura popular, <head>arte</head> y tradición".*

Input Tagged sentence:
*<w frm="cultura" lem="cultura" pos="NCFS000"/>*
*<w frm="popular" lem="popular" pos="AQ0CS0"/>*
*<w frm="," lem="," pos="Fc"/>*
*<w frm="arte" lem="arte" pos="NCCS000" head="yes"/>*
*<w frm="y" lem="y" pos="CC"/>*
*<w frm="tradición" lem="tradición" pos="NCFS000"/>*

Output:
*cultura NCFS000, popular AQ0CS0, arte NCCS000, y CC, tradicion NCFS000*

Figure 2 Sample Sentence

Next, a file with training examples for each word is fed to the BoosTexter system. Once training is completed, a file with the examples from the test set for each word is fed to the BoosTexter system. Lastly, the individual test set files are concatenated to be evaluated.

### 4.1 Scoring Scheme

Evaluation of the results has been carried out by SENSEVAL-3 using an official scorer proposed by Resnik and Yarowsky [19] and two other evaluations for the particular task. The official scorer is derived by assigning probabilities over sense labels generated by WSD algorithms. Given a probability distribution over sense labels and a single known-correct sense label, the algorithm's score should be the probability that the algorithm assigns to the correct sense label. One of the other scoring schemes does a more complete evaluation, including word-by-word results and results by groups of words. Words are grouped by part-of-speech, i.e., noun (n), verb (v), adjective (a), and by the accuracy of the Most-Frequent-Sense baseline classifier. The third scoring scheme takes the results of a baseline system that always assigns to each word the most frequent sense according to the training set.

## 5 Experiments

In our experiments we work with 46 datasets, one per word to be trained. The number of classes, i.e., senses ranges from 2 to 8, the number of training examples ranges from 69 to 268; the number of examples in the test set is over half the number of training examples (see Table 1).

| Word | POS | Sen ses | Train Ex. | Test Ex. | Word | POS | Sen ses | Train Ex. | Test Ex. | Word | POS | Sen ses | Train Ex. | Test Ex. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| actuar | v | 3 | 133 | 67 | corona | n | 3 | 124 | 64 | partido | n | 2 | 133 | 66 |
| apoyar | v | 3 | 259 | 128 | duplicar | v | 2 | 254 | 126 | pasaje | n | 4 | 220 | 111 |
| apuntar | v | 4 | 213 | 106 | explotar | v | 5 | 212 | 103 | perder | v | 4 | 218 | 106 |
| arte | n | 3 | 251 | 121 | ganar | v | 3 | 237 | 118 | popular | a | 3 | 133 | 67 |
| autoridad | n | 2 | 268 | 132 | gracia | n | 3 | 72 | 38 | programa | n | 3 | 267 | 133 |
| bajar | v | 3 | 235 | 115 | grano | n | 3 | 117 | 61 | saltar | v | 8 | 200 | 101 |
| banda | n | 4 | 230 | 114 | hermano | n | 2 | 128 | 66 | subir | v | 3 | 231 | 114 |
| brillante | a | 2 | 126 | 63 | jugar | v | 3 | 236 | 117 | simple | a | 3 | 117 | 61 |
| canal | n | 4 | 262 | 131 | letra | n | 5 | 226 | 114 | tabla | n | 3 | 130 | 64 |
| canalizar | v | 2 | 253 | 126 | masa | n | 3 | 172 | 85 | tocar | v | 6 | 158 | 78 |
| ciego | a | 3 | 102 | 52 | mina | n | 2 | 134 | 66 | tratar | v | 3 | 143 | 72 |
| circuito | n | 4 | 261 | 132 | natural | a | 5 | 215 | 107 | usar | v | 2 | 263 | 130 |
| columna | n | 7 | 129 | 64 | naturaleza | n | 3 | 258 | 128 | vencer | v | 3 | 134 | 65 |
| conducir | v | 4 | 134 | 66 | operación | n | 3 | 134 | 66 | verde | a | 2 | 69 | 33 |
| corazón | n | 3 | 123 | 62 | órgano | n | 2 | 263 | 131 | vital | a | 2 | 131 | 65 |
| | | | | | | | | | | volar | v | 3 | 122 | 60 |

Table 1 Set of 46 words

The total number of training examples used was 8430, and the total number of test examples was 4195.

The binary-valued attributes for describing the examples correspond to five features which constitute a very narrow linguistic context. The five features which are a modification of those used in [10] are $l_{i-2}$ $pos_{i-2}$, $l_{i-1}$ $pos_{i-1}$, $l_i$ $pos_i$, $l_{i+1}$ $pos_{i+1}$, $l_{i+2}$ $pos_{i+2}$, $l$ is a lemma and $pos$ is its corresponding part-of-speech.

We trained the system using BoosTexter for each word a different number of iterations depending on the number of examples for a word in the training set and the different senses attributable to the word. The assumptions made are: the more examples to be trained, the more attributes need to be examined to determine a weak rule and the more senses for a word the higher the probability of a word to be misclassified. The total number of iterations and the number of hypotheses are generated per word during the learning process. Once the training was completed we ran Boostexter on the test set. The results generated were submitted to SENSEVAL-3 for evaluation.

Overall results of the official evaluation, the POS-based evaluation, and the Most Frequent Sense (MFS) evaluation for the test set are shown respectively in Table 2, Table 3, and Table 4.

| | | |
|---|---|---|
| precision | 74.09% | 3108 correct of 4195 predictions |
| recall | 74.09% | 3108 correct of 4195 in total |
| F1 score | 74.09 | F1 = (2*precision*recall)/(precision+recall) |
| coverage | 100.00% | 4195 examples predicted of 4195 in total |

Table 2 Overall Results

| POS | total | predicted | hit | coverage | precision | recall | F1 |
|---|---|---|---|---|---|---|---|
| a | 448 | 448 | 347 | 100.00 | 77.46% | 77.46% | 77.46 |
| n | 1949 | 1949 | 1440 | 100.00 | 73.88% | 73.88% | 73.88 |
| v | 1798 | 1798 | 1321 | 100.00 | 73.47% | 73.47% | 73.47 |

Table 3 POS-based Average Evaluation

| word-group | total | predicted | hit | coverage | precision | recall | F1 |
|---|---|---|---|---|---|---|---|
| 1.MFS>95 | 635 | 635 | 608 | 100.00 | 95.75% | 95.75% | 95.75 |
| 2.MFS90-95 | 429 | 429 | 394 | 100.00 | 91.84% | 91.84% | 91.84 |
| 3.MFS80-90 | 374 | 374 | 325 | 100.00 | 86.90% | 86.90% | 86.90 |
| 4.MFS70-80 | 618 | 618 | 394 | 100.00 | 63.75% | 63.75% | 63.75 |
| 5.MFS60-70 | 523 | 523 | 361 | 100.00 | 69.02% | 69.02% | 69.02 |
| 6.MFS50-60 | 586 | 586 | 390 | 100.00 | 66.55% | 66.55% | 66.55 |
| 7.MFS40-50 | 673 | 673 | 456 | 100.00 | 67.76% | 67.76% | 67.76 |
| 8.MFS<40 | 357 | 357 | 180 | 100.00 | 50.42% | 50.42% | 50.42 |

Table 4 Most Frequent Sense

Analysis of individual results for each word show that the number of features chosen for training needs to be increased to include more features. Additional syntactic features and/or prior knowledge are necessary to improve recall and precision of some words such as the nouns *letra, columna, gracia* and verbs such as *perder, conducir, tocar, saltar*. In the case of the nouns *letra* and *columna*, the number of senses is 5 and 7 respectively, which makes the disambiguation task more difficult; but it also may be the case that the training examples given do not contribute to generate a highly accurate hypothesis. A similar situation occurs with the verbs *saltar* with 8 senses, *tocar* with 6 senses and *perder* and *conducir* with 4 senses each. Results including the top five and the bottom five are presented in Table 5.

| word | total | predicted | hit | coverage | precision | recall | F1 |
|---|---|---|---|---|---|---|---|
| usar.v | 130 | 130 | 127 | 100.00 | 97.69% | 97.69% | 97.69 |
| canalizar.v | 126 | 126 | 122 | 100.00 | 96.83% | 96.83% | 96.83 |
| autoridad.n | 132 | 132 | 127 | 100.00 | 96.21% | 96.21% | 96.21 |
| duplicar.v | 126 | 126 | 121 | 100.00 | 96.03% | 96.03% | 96.03 |
| hermano.n | 66 | 66 | 62 | 100.00 | 93.94% | 93.94% | 93.94 |
| conducir.v | 66 | 66 | 36 | 100.00 | 54.55% | 54.55% | 54.55 |
| gracia.n | 38 | 38 | 19 | 100.00 | 50.00% | 50.00% | 50.00 |
| columna.n | 64 | 64 | 31 | 100.00 | 48.44% | 48.44% | 48.44 |
| perder.v | 106 | 106 | 50 | 100.00 | 47.17% | 47.17% | 47.17 |
| letra.n | 114 | 114 | 50 | 100.00 | 43.86% | 43.86% | 43.86 |

Table 5 Results for 10 words

## 6 Related Work

Some of the supervised learning approaches applied in the Spanish Lexical Sample task were an exemplar based classifier [20], support vector machines [21], decision trees [24], pattern abstraction, and kernel methods [22] and a combination of three classifiers [23]. The use of these approaches in the Spanish Lexical Sample Task is briefly described next.

The exemplar-based classifier measures the similarity between a new instance and the representation of some labelled examples. The terms are represented as bags of contexts. Words, lemmas and senses are represented in the same space, called Context Space, where similarity measures can be defined. In the SVM approach each training and test item is represented as a feature with weights; its dimensions correspond to properties of the context. A family of SVM classifiers was constructed for the senses of each word. All positive training examples for a word sense were treated as negative examples for all the other senses. The decision trees approach uses an ensemble of three bagged decision trees. It is based on the idea that different views of the training examples for a given target word will result in classifiers that make complementary errors. Thus their combined performance will be better than individual performances. Pattern abstraction is

a methodology which uses different knowledge sources to extract information. This represents a limitation that has been solved with kernel methods. Kernels are similarity functions between instances that allow the integration of different knowledge sources and the modelling of linguistic features in SVM. The combination of classifiers included a nearest-neighbor clustering classifier, a naive Bayes classifier, and a decision list classifier; each one was trained on several permutations of the extracted feature set, then the answers were combined using voting.

## 7 Conclusion and Future Work

Our precision, recall and $F_1$ measure were very close to the Most Frequent sense Classifier (MFC) scores reported by the task organizer. Compared with the supervised learning techniques presented, the boosting algorithm applied to WSD of Spanish did not perform well. We found some inconsistencies in the results for some words and are repeating experiments for those words. We have no conclusive results to report in this paper due to time constraints. Compared with unsupervised learning techniques, the boosting algorithm performed better. A detailed description of the results and system comparisons appears in [25]. Further research includes the following tasks: testing the algorithms in other Spanish tagged corpora; implementing, comparing and evaluating other supervised learning approaches; and adding syntactic features.

## References

1. Schapire, R., Singer, Y.: BoosTexter:A Boosting-based System for Text Categorization *Machine Learning*, **38**(2/3):135-168, 2000.
2. Ide, N., Veéronis, J.: Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art. *Computational Linguistics*, **24**(1):1-41, 1998.
3. Schapire, R.E.: The Boosting Approach to Machine Learning An Overview. *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.
4. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, **55**(1):119-139, August,1997.
5. Schapire, R.E., Singer, Y.: Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, **37**(3):297-336,1999.
6. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, 148-156, 1996.
7. Bauer, E., Kohavi, R.: An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants. *Machine Learning Journal. Special Issue on IMLM for Improving and Scaling Machine Learning Algorithms*, **36**(1-2):105-139,July/August,1999.
8. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139-158.
9. Quinlan, J. R.: Bagging, boosting and C4.5. *Proceedings of the 13th National Conference on Artificial Intelligence, AAAI*, 1996.

10. Escudero, G., Màrquez, L., Rigau, G.: Boosting Applied to Word Sense Disambiguation. *Proceedings of the 12th European Conference on Machine Learning, ECML 2000*, Barcelona, Spain.
11. Schapire, R.E.: Advances in Boosting. *Uncertainty in Artificial Intelligence: Proceedings of the Eighteenth Conference*, 2002.
12. Rochery, M., Schapire, R., Rahim, M., Gupta, N., Riccardi, G., Bangalore, S., Alshawi, H., Douglas, S. : Combining Prior Knowledge and Boosting for Call Classification in Spoken Language Dialogue. *Intrnational Conference on Acoustics, Speech and Signal Processing*, 2002.
13. Abney, S., Schapire, R.E., Singer, Y.: Boosting Applied to Tagging and PP Attachment. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.
14. Iyer, R., Lewis, D., Schapire, R.E., Singer, Y., Singhal, A. Boosting for Document Routing. *Ninth International Conference on Information and Knowledge Management*, 2000.
15. Fujii, A., Inui, K., Tokunaga, T., Tanaka, H.: Selective Sampling for Example-based Word Sense Disambiguation. *Computational Linguistics*, **24**(4):573-598, 1998.
16. Yarowsky, D.: Decision Lists for Lexical Ambiguity Resolution: Application ro accent restoration in Spanish and French. *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, ACL*, 1994.
17. Towell, G., Vorhees, E.M.: Disambiguating Highly Ambiguous Words. *Computational Linguistics*, **24**(1):125-145, 1998.
18. SENSEVAL-3: URL *http://www.senseval.org/senseval3*
19. Resnik, P., Yarowsky, D.: A perspective on word sense disambiguation methods and their evaluation, position paper presented at the *ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, 1997
20. Artiles, J., Peãs, A., Verdejo, F.: Word Sense Disambiguation based on Term to Term Similarity in a Context Space. *ACL SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004
21. Cabezas, C., Battacharya, I., Reskink, P.: The University of Maryland SENSEVAL-3 System Descriptions. *ACL SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.
22. Strapparava, C., Gliozzo, A., Giuliano, C.: Pattern Abstraction and Term Similarity for Word Sense Disambiguation: IRST at Senseval-3. *ACL SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.
23. Wicentowski, R., Thomforde, E., Packel, A.: The Swarthmore College SENSEVAL3 System. *ACL SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.
24. Pedersen, T.: The Duluth Lexical Sample Systems in SENSEVAL-3: *ACL SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.
25. Màrquez, L., Taulé, M., Martí, M.A., García, M., Artigas, N., Real, F.J., Ferrés, D. SENSEVAL-3: The Spanish Lexical Sample Task. *ACL SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.

# A Cognitive-Based Approach to Adaptive Intelligent Multiagent Applications

Charles Hannon

Department of Computer Science
Texas Christian University
Fort Worth, TX
c.hannon@tcu.edu

**Abstract.** An integrated cognitive-based model (LEAP) and application (SALT) are presented. Building on a new Interlaced Micro-Patterns (IMP) theory and the Alchemy/Goal Mind environment, the LEAP research improves agent-to-human and agent-to-agent communication by incorporating aspects of human language development. The IMP theory further provides a theoretical basis for deep incorporation and sharing of knowledge from different sensor modalities. Research on LEAP points to a better understanding of human language development and the application of this knowledge within intelligent multiagent applications Research with SALT points to how this research supports Smart Home applications and provides feedback to LEAP modeling.

## 1 Introduction

Many intelligent multiagent applications can be improved by an adaptive agent organization that can not only re-task existing agents, but also add new agent capabilities to deal with changing requirements. While this level of agent adaptability presents a complex problem in design and construction, humans present an archetype for such abilities. In this article we will show how a study of one complex human skill (reading) can be used to drive adaptive multiagent design and how the information used from this study can be used in a Smart Home multiagent application.

To study language use and learning within a reading task, a robust distributed cognitive model called LEAP (Language Extraction from Arbitrary Prose) and a new working theory of cognition called IMP (Interlaced Micro-Patterns) are used. One focus of the LEAP/IMP research is to study how lexical, syntactic, semantic and conceptual information can be learned from a set of English language web-based sources. However, LEAP can also explain how language development occurs within the context of general cognitive development using all sensory modalities. By focusing on both ability and performance within this broader context, LEAP can provide insight into more general use and learning of cognitive skills that can be directly integrated into intelligent multiagent applications that serve to test the current working theories (e.g., IMP) of the models themselves.


*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 239-248*

LEAP is developed using the components of the Gold Seekers project depicted in Figure 1. Building on existing non-computational models and other research, the Gold Seekers project attempts to develop working theories (like IMP) that can be used to build modular computational models using Alchemy/Goal Mind [7]. These modules (or Agent Components) can be reused in other agent models to test other aspects of cognition or as the starting point of cognitive-based applications. The Smart-environment Adaptive-agent Language and Tasking (SALT) application builds on the our model research to explore how a dynamically constructed and tasked multiagent model can be used to allow a smart environment to better adapt to its users. SALT forms an integral part of the overall research by providing feedback on how the models handle a 'real world' application.

## 2 Related Work

Numerous ongoing research projects have applied machine learning techniques directly to the way a smart environment learns user preferences. The SALT application research is focused on adaptation through the way agents communicate and share tasks. For this reason we will focus our related work discussion on the LEAP model.

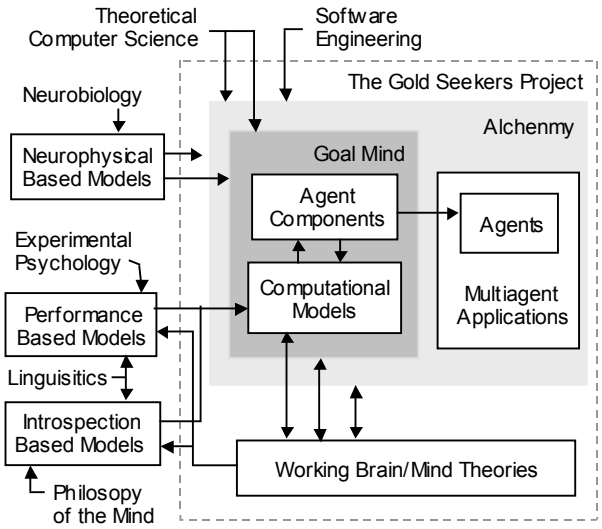



**Fig. 1.** Computational models are used to produce the agent components making up agents within a multiagent application. Components' design, construction, testing and operation are supported by Goal Mind. The distribution, migration and control of component processes and the resulting agent multi-processes across multiple processors are supported by Alchemy.

The two language models that LEAP directly build upon are the TALLUS [5] and STRESS [6]. We will briefly contrast the current research with some other cognitive modeling environments and two language-related knowledge bases, before addressing the theory of spreading activation which is a key element in the LEAP model.

### 2.1 Related Language Modeling and Capture Efforts

A number of on-going research efforts are addressing the cognitive modeling of language at some level. Many of these models address language within the context of other sensor modalities and are aimed at directly supporting an agent-based application. LEAP attempts to; 1) be explanatory, 2) be closely tied to well know cognitive mechanisms such as priming, spreading activation and memory consolidation, and 3) directly support use of its components within multiagent applications. This makes it similar to models built with SOAR [8], ACT-R [1] and ACT-R/PM [3]. The major difference between Alchemy/Goal Mind and these other architectures is that the Alchemy/Goal Mind models are created out of a set of concurrent components which are free to use their own cognitive sub-theories with the main cognitive theory controlling the method in which these components interact. This can be contrasted with the other environments where models are monolithic processes controlled some underlying cognitive mechanism such as ACT-R's symbolic productions and subsymbolic activations.

Some multiagent efforts rely on existing language knowledge bases. Compared to projects like WordNet and Cyc, that attempt to capture language and concept knowledge in large publicly available databases, LEAP currently has an extremely small database of language knowledge. For example, WordNet contains 144,309 unique words organized into synonym sets representing underlying lexical concepts [4]. The Cyc knowledge base contains almost 2 million assertions (rule and fact), 118,000 concepts and a lexicon of 17,000 English root words [11]. Both WordNet and Cyc have been very instrumental in our discovery of the underlying structure of the way language and concept reasoning works, but this does not directly translate to making them useful candidates for knowledge components within an adaptive multiagent application. In contrast to simply using a vast store of language knowledge, LEAP is attempting to capture the way humans learn by the slow consolidation of knowledge into a complex and multifaceted representation of their surrounding world and to use the resulting structure of that representation to discover how we can simulate human development within adaptive agents.

### 2.2 Spreading Activation

Memory priming via a spreading activation mechanism is a very old concept going all the way back to a direct extension of the Quillian work on Semantic Memory in the 1960's [10]. The original theory, proposed by Collins and Lofus in 1975 proposes that is-a, reverse is-a (what TALLUS and LEAP calls a could-be relation), has-a and part-of semantic relations will be followed to activate parent, children, and other associated nodes within a person's semantic network making it easier to retrieve these

concepts immediately after the original concept retrieval [9]. Several psychology experiments in the 1970's and later demonstrated that the priming effects proposed by the spreading activation mechanism were observable [2].

While symbolic AI systems have focused mostly on the priming aspects of spreading activation, a number of connectionist systems have also explored the effect of lateral inhibition where the activation of a concept can cause the retrieval of related concepts to be blocked for a period of time after that activation. Psychology experiments appear to show that lateral inhibition works with spreading activation to allow us to more quickly determine that some statements are counter-to-fact [10].

The most mature cognitive modeling treatment of spreading activation to date is seen in the ACT-R architecture which uses production rules, not a classic semantic network, in its primary knowledge representation. Other semantic network based systems that use it tend to do something similar to ACT-R by calculating the amount of activation for a node based on its distance from the activated node, and then using the resulting number to artificially control the lookup between nodes during inference. As will be shown later, we will take a radically different approach to simulation spreading activation in the LEAP model.

## 3 Interlaced Micro-Patterns (IMP) Theory

Pattern matching as an important mechanism in the learning, retrieval and recall of simple concepts and procedures have been accepted in both machine learning and cognitive psychology research for some time. The Interlaced Micro-Patterns (IMP) cognitive theory extends the traditional pattern matching mechanism by proposing that if a set of simple patterns are interlaced (i.e., allowed to overlap), the mechanism can be used to learn, retrieve and recall elements of far greater complexity, and thus, could be the driving mechanism of such tasks as language use and learning. The support for IMP as a working theory comes from both a set of thought problems and the results of cognitive modeling work.

The first language model using what would become Alchemy/Gold Mind was TALLUS which was designed to study telegraphic speech (the second true language development phase in humans) within a visual context. Like most language models, TALLUS used a standard generative linguistic theory that proposed that utterances are generated by phrase structure rules that result in the utterance being associated as the leaf nodes of a hierarchical tree structure starting from a root node utterance or sentence. Given a set of generative rules, TALLUS could easily learn new surface forms and their associated concepts, but no believable explanatory mechanism for learning new syntax and their associated conceptual grids could be found.

This model failure resulted in the first thought problem. Why do children find it much easier to learn a natural language than the proposed grammar rules that are suggested to define such a language? Hierarchical syntactic approaches to natural language (NL) align well with the way NL grammars are taught in traditional educational settings, but not with how language development naturally occurs. The teaching of prescriptive grammars may help to stabilize language use across a group of language users, but it seldom controls the complete use of either spoken or written

language 'rules' in that group with much of that use being driven by either a conscious or unconscious violation of the prescriptive rules. Many non-generative linguistic theories use this same argument to dismiss generative approaches, but these theories seldom provide a mechanism that could be used in a computational model of language.

So, is there a way to capture the computational strength of generative grammar without it being driven by a hierarchical set of rules? One possible method to do this is to use interlaced micro-patterns. While all possible well-formed utterances conform to some syntactic, semantic and conceptual pattern, the storage of every possible utterance pattern would clearly be too computationally complex to be feasible. However, if all possible sentence patterns were made up of smaller patterns that relied on overlapping elements to ensure correctness, a set of smaller patterns could not only generate correct utterances, but also block the creation of malformed utterances.

To test this approach, the LEAP model was constructed, which has confirmed the viability of the IMP theory for language learning. Further, it has introduced two new questions. Could the IMP theory supply an underlying mechanism for all cognition? And, could differences in the potential size of micro-patterns and their ability to interlace be an underlying control in the level of cognitive abilities exhibited by a biological organism?

### 3.1 IMPs Relationship to Symbolic AI

It is fairly simple to see how the IMP theory would map to a connectionist approach since the patterns can simply be distributed among the weights of connections; however, Alchemy/Gold Mind is basically a symbolic AI approach so we need to address the symbolic mapping a little further. Due to the large amount of existing research with different Knowledge Representation and Reasoning (KRR) methods, what we do not want is a theory that limits the types of symbolic reasoning possible within an application. Luckily, it can be shown that using the IMP theory as an overall control mechanism does not require such a limitation.

In summary, we can define a system's composite KRR as a set of layered component KRRs with each component's KRR being any desired type. This composite KRR can be stored in Long Term Memory (LTM) and access points within each layer can be activated into Short Term Memory (STM) by a pattern input from an external source (either another layer within the agent or an interface to the external world). In addition to the actual access points activated, other parts of the layer's KR can be activated by a temporal-based spreading activation mechanism when needed and deactivated by removal from the STM when the knowledge is 'timed-out'. Changes to the KRR can occur by updating the KR stored in LTM as a result of changes that occur in STM during activation.

A simple formalization of the effect of using IMP to control a layered KRR can be given if we simplify the KR of an agent to a uniform set of semantic networks. Each of these semantic networks can be viewed as a directed multi-graph,

$$R_n = \text{pair}\,(N_n, A_n),\ A_n = \{(\nu_{ni}, \nu_{nj}) \mid \nu_{ni}, \nu_{nj} \in N_n\} \quad \textbf{(1)}$$

where, n is the level of representation, $N_n$ is a set of nodes, and $A_n$ is a bag of named relationships between these nodes. A sub-representation of this network can be defined as,

$$R'_n = \text{pair}\,(N'_n, A'_n),\ N'_n \subseteq N_n, \text{ and} \quad (2)$$
$$A'_n \subseteq A_n \wedge ((\nu_{ni}, \nu_{nj}) \in A'_n \rightarrow \nu_{ni}, \nu_{nj} \in N'_n).$$

All possible sub-representations at a level n is, of course, the power set of $R_n$; however, this set has little meaning in the IMP theory since only the activated subrepresentations are of interest. Given all possible activated sub-representations at a level n, defined as,

$$\Phi_n = \{R'_n \mid R'_n \subset R_n \wedge \text{active}(R'_n) \rightarrow \text{True}\}, \quad (3)$$

connections between representation levels can also be viewed as a directed multigraph,

$$K_{i,j} = \text{pair}\,(\Phi_{i,j}, \Gamma_{i,j}),\ \Phi_{i,j} = R'_i \cup R'_j, \text{ and} \quad (4)$$
$$\Gamma_{i,j} = \{(R'_i, R'_j) \mid R'_i, R'_j \in \Phi_{i,j}\},$$

where, i and j are levels of representation being connected and $\Gamma_{i,j}$ is a set of named relationships between these levels.

The number of representation levels ($R_n$) and number of level connections ($K_{i,j}$) can vary based on application. A traditional agent-based method for using the overall representation structure would be a set of m stacks of representation levels 1 to k with the top-level (level 1) of each stack being an interface representation and the kth level of each stack being either a common conceptual structure or a set of connected conceptual structures.

Given a set of available general inference rules at each level ($\rho_n$) and between two levels ($\rho_{i,j}$), the extent of general inference at each level ($\iota_n$) and across levels ($\iota_{i,j}$) can be naively described as,

$$\iota_n \approx |\,\rho_n\,| \text{ and } \iota_{i,j} \approx |\,\rho_{i,j}\,|, \quad (5)$$

assuming no serious difference exist in the number of pre and post conditions of each rule. The total extent of representation at each level also can be naively described as,

$$\gamma_n \approx |\,N_n\,| \bullet \max\,\{\nu_{ni}, \nu_{nj}\} d(\nu_{ni}, \nu_{nj}), \quad (6)$$

which given the amount of accessible (or activated) knowledge at each level being $\beta_n = \cup\Phi_n$, leads to an activated representation extent of,

$$\alpha_n \approx |\,\beta_n\,| \bullet \max\,\{\nu_{ni}, \nu_{nj}\} d(\nu_{ni}, \nu_{nj}) \mid \nu_{ni}, \nu_{nj} \in \beta_n \text{ and} \quad (7)$$
$$\alpha_{i,j} \approx |\,\Phi_{i,j}\,| \bullet \max\,\{R'_i, R'_j\} d(R'_i, R'_j).$$

The activation potential at any level can be described as,

$$\eta_n \approx \Sigma_{\{i = 1 \text{ to } k\}}\ \alpha_{i,j} \bullet \iota_{i,j}, \quad (8)$$

and its inference potential as,

$$\kappa_n \approx \alpha_n \bullet \iota_n \,. \tag{9}$$

Assuming that we only allow a pattern matching activation mechanism to work between levels, the extent of cross-layer general inference ($\iota_{i,j}$) can be viewed as approaching the value one for all levels. Thus, the activation potential of all levels becomes approximately equal to their part of the cross-layer activated representation extent, $\alpha_{i,j}$, which is simply their own activated representation extent $\alpha_n$. Thus, a pattern matching interface between layers reduces the overall inference potential in each layer to a function of the number of activated access points and its own inference extent. To the outside world, any results of a layer's inference engine look like an Artificial Neural Network's (ANN) forward or backward activation potentials.

## 4 The LEAP Model

The LEAP model is a distributed model for learning lexical, syntactic, semantic and conceptual information about English from web-based sources. It is currently made up of twenty Goal Mind components (each a multithreaded LINUX process) built on the environment's production system and semantic network libraries and its standard PostgreSQL 'C' language interface.

At the surface language layer, LEAP uses a set of seven lexical analyzers and a special purpose Stimuli Routing Network (SRN) used to filter some closed categories. It can discover new instances of open part-of-speech (PoS) categories and new patterns of word use within the input utterances. The concept reasoner's Situational Dependences Semantic Network (SDSM) [5] uses spreading activation to allow a very large network to exist in compressed form in the PostgreSQL database (simulating Long Term Memory or LTM) while small pieces of the network can be uncompressed into a dynamic memory structure within each of the concept reasoners (simulating Short Term Memory or STM).

When a word comes in from the models HMI or HTML reader, all lexical analyzers look up the word in their PoS form table and send either an active or inhibit PoS stimulus message based on this lookup. If the word is not found (i.e., either it is not in the PoS form table or has too low a belief to be used), an analyzer uses reports from other analyzers to try to find a PoS pattern in its PoS pattern table that would indicate that the word may be of its PoS type. If a pattern is found, the word is either added to the PoS form table with a very low belief or the belief of the existing form is incremented based on this example that the word matches the expected pattern. If the word is found but the surrounding words' PoS do not match an existing pattern, a pattern is either added to PoS pattern table with a very low belief or the belief of the existing pattern is incremented based on this example of a valid pattern.

When a concept is looked up, it is copied from the database (LTM) to the dynamic memory (STM) of a concept reasoner and given the maximum time to live by setting its countdown timer to the maximum allowed value. In addition, all other nodes connected via a set number of outbound relations are also activated (moved to memory) and given a time to live based on their distance from the concept that was directly

accessed. The concept reasoner is only allowed to inference across active nodes, but when it makes a valid connection between two concepts, it both reports the finding and resets the time to live values for all nodes in the inference path. Running in the background of each concept reasoner is a temporal collector that decrements the countdown timer of each node during each time-slice and removes any node whose counter alarms (hits zero) from STM.

## 5 The SALT Application

Building on the LEAP and earlier models, the SALT application explores how to dynamically construct and task the control system for a smart environment. It has long been recognized that smart environments need to learn the preferences of their users, but to move them from the lab to mainstream use they will also need to adapt to different and changing hardware environments. Each instance of a smart environment will need to fit into an unique location where size, cost and other factors will determine the hardware being used. These systems will need to be able to accept new smart components as they become available. Further, the control system must be able to 'work around' failed hardware components to ensure both user comfort and safety.

In the current SALT application, five agents are used to test how these agents can learn to communicate and distribute system tasking using a simplified language based on human language lexical, syntactic and semantic constructs. Using seventy-two Goal Mind components, the model current focuses more on language use than the interface to system hardware or the smart environment control, but past Goal Mind research indicates that more robust hardware interfaces and control structures can be added by less than doubling the number of processes in these agents. Running an application with about hundred and fifty processes is well within the capacity of Alchemy to handle on a relatively small Beowulf cluster.

In the current SALT model, a Control and Human Interface agent provides both the Human Machine Interface (HMI) and overall smart environment task distribution. A Kitchen agent controls kitchen appliances and provides meal planning and food inventory control. An Entertainment agent controls entertainment equipment and provides setup based on user preference. An Environmental Control agent monitors A/C and safety components and attempts to match user preferences to safety and efficiency constraints. An Inhabitant and Robot Tracking agent provides the system with situation awareness about mobile elements of the environment using a multiple-camera-based vision system.

Many aspects of the SALT application can exploit the power of the IMP theory. A direct application of the LEAP research in SALT is in its HMI. By integrating the HMI directly to the agents' other sensor modalities, the resulting language interface can be very adaptive. In the future we will build on this to allow the dynamic allocations of agents within the system to support different environments and hardware conditions.

## 6 Initial Results and Future Work

Both the LEAP and SALT research are presented here to provide an overview of how the Gold Seekers environment allows the meaningful integration of both the theory and application of cognitive approaches. To date, the LEAP results are better understood and will be the focus of this section.

Reading tests with the LEAP model have been conducted on a number of children's stories and web-based news articles. Our current focus is on the size of micro-patterns needed for LEAP to support surface and deep structure language learning and use. To support testing, the learning results, which are stored in a PostgreSQL database are compared against a version of WordNet also store in a PostgreSQL database.

Current LEAP results fall into two basic categories, detailed analysis of individual readings and general observations about language development and the reading task. As expected, the current data from news articles shows that a great deal more language priming is needed to learn at the same rate as with children's stories. These have led to several general observations. First, that using IMP, there is no good way to short-circuit the normal development process starting with simple stories and work up to more and more complex articles. Second, that reading development needs input from other sensor modalities.

Most of the work with children's stories demonstrate similar results so as an example of a detailed analysis we will focus on a single story, *Robert the Rose Horse*. The story contains approximately 1100 words and 200 utterances. This gives a mean-length-of-utterance (MLU) of about 5.5. Ignoring closed categories, the vocabulary is about 90 words. From this and other children's stories studied, it is clear that authors focus on the reduction of word length, lexical complexity and the MLU, but do not necessarily attempt to reduce the syntactic complexity of the resulting utterances.

Using a database primed with 15 words of the core vocabulary and no patterns, LEAP was able to detect 26 patterns. Increasing the core vocabulary to 20 words by adding 5 nouns produced one additional noun pattern, while increasing the core vocabulary to 22 words by adding 2 verbs produced 10 additional patterns. In all cases the patterns were non-conflicting between PoSs. Most patterns show a minimal amount of repeatability within a story, but a few are highly repeatable due to the prose structure of children's stories. These differences in repeatability is not common in news articles. Continuing to add nouns and verbs to the core vocabulary continues to show the same data trend where verbs influence the number of patterns found more than other PoSs. As a result of the spreading activation mechanism used in the concept reasoner, associations between new and exiting concepts can be more easily identified. In *Robert the Rose Horse* this method was used to learn that 'rose' is a type of 'flower' and 'bank' is a thing that can be 'robbed'.

Applying LEAP results to SALT is driven by earlier work with the TALLUS model. In TALLUS, we were able to greatly simplify the language generation task by focusing on telegraphic speech patterns. The agent communication in SALT currently relies on the same reduction in language complexity. There is clearly a point where using a non-formal adaptive language adds too much overhead to an overall agent, but a SALT-like environment is targeted for highly intelligent agents where this is not

a major factor. Initial work with SALT has shown that given a small shared vocabulary and a set of patterns that represent a simple syntax, agents can learn a shared language. We are still working on each agent's semantic ties to this language.

Both the Gold Seekers' toolset and the current models support the ability to dynamically create new agents which would allow a SALT-like application to add new agents as new hardware is added and support other changes to the overall environment. While this is clearly an interesting line of research, the short-term focus of both LEAP and SALT is in improving the integration of their language use.

## 7 Conclusion

Current work with IMP, LEAP and SALT demonstrate that they are providing valuable information about human language development and adaptive agent communication. As other uses of the IMP theory are explored, it is hoped that it will provide a general mechanism for adaptive intelligence within a multiagent environment. The SALT-based research should continue to provide an even better platform for testing the concepts proposed by the LEAP research. While the integration of the LEAP and SALT research paths provide complex challenges, the result of such integration appears to be worth such complexity.

## References

1. Anderson, J. R. and Lebiere, C. *Atomic Components of Thought*. Hillsdale, NJ: Lawrence Erlbaum Associates, Pub., 1998.
2. Anderson, J. R. *Cognitive Psychology and its Implications*. New York: W. H. Freeman and Company, 1995.
3. Byrne, M., "ACT-R/PM and Menu Selection: Applying a Cognitive Architecture to HCI", *International Journal of Human Computer Studies*, 1999.
4. Fellbaum, C (Editor), *WordNet: An Electronic Lexical Database*, Cambridge, MA, MIT Press, 1998.
5. Hannon, C. and D. J. Cook. "Developing a Tool for Unified Cognitive Modeling using a Model of Learning and Understanding in Young Children." *The International Journal of Artificial Intelligence Tools*, 10 (2001): 39-63.
6. Hannon C. and D. J. Cook. "Exploring the use of Cognitive Models in AI Applications using the Stroop Effect." Proceedings of the Fourteenth International Florida AI Research Society Conference, May 2001.
7. Hannon, C., A Geographically Distributed Processing Environment for Intelligent Systems. *In Proceedings of PDPS-2002*. 355-360, 2002.
8. Newell, A. *Unified Theories of Cognition*. London: Harvard University Press, 1990.
9. Martindale, C., Cognitive Psychology: A neural-Network Approach, Belmont, CA, Brooks/Cole, 1991.
10. Smith, G. W., *Computers and Human Language*, New York: Oxford Press, 1991.
11. Witbrock, Michael, D. Baxter, J. Curtis, et al. An Interactive Dialogue System for Knowledge Acquisition in Cyc. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003.

# Modeling and Intelligent Controland Logic

# Orthogonal-Back Propagation Hybrid Learning Algorithm for Interval Type-2 Non-Singleton Type-2 Fuzzy Logic Systems

Gerardo M. Mendez[1], Luis A. Leduc[2]

[1] Department of Electronics and Electromechanical Engineering
Instituto Tecnológico de Nuevo León
Av. Eloy Cavazos #2001, Cd. Guadalupe, NL, CP. 67170
MÉXICO
gmmendez@itnl.edu.mx
[2] Department of Process Engineering
Hylsa, S.A. de C.V.
Monterrey, NL.
MÉXICO
lleduc@hylsamex.com.mx

**Abstract.** This article presents a new learning methodology based on an hybrid algorithm for interval type-2 non-singleton type-2 fuzzy logic systems (FLS) parameters estimation. Using input-output data pairs during the forward pass of the training process, the interval type-2 FLS output is calculated and the consequent parameters are estimated by orthogonal least-square (OLS) method. In the backward pass, the error propagates backward, and the antecedent parameters are estimated by back-propagation (BP) method. The proposed hybrid methodology was used to construct an interval type-2 fuzzy model capable of approximate the behavior of the steel strip temperature as it is being rolled in an industrial Hot Strip Mill (HSM) and used to predict the transfer bar surface temperature at finishing Scale Breaker (SB) entry zone. Comparative results show the advantage of the hybrid learning method (OLS-BP) over that with only BP.

## 1 Introduction

Interval type-2 fuzzy logic systems (FLS) constitute an emerging technology. In [1] the interval type-2 FLS learning methods are one-pass and back-propagation (BP) methods. One-pass method generates a set of IF-THEN rules by using the given training data once, and combines the rules to construct the final FLS. In back-propagation, none of antecedent and consequent parameters of the interval type-2 FLS are fixed at starting of training process; they are tuned using BP method. Recursive least-square (RLS) is not presented as interval type-2 FLS learning method.

One-pass and Back-Propagation (BP) are presented as type-2 FLS learning methods in [1]. One-pass method generates a set of IF-THEN rules by using the given

© A. Gelbukh, R. Monroy. (Eds.)
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 251-260*

training data once, and combines the rules to construct the final fuzzy logic systems (FLS). None of the antecedent and consequent parameters of interval type-2 FLS are fixed at the start of the training process in BP; instead they are tuned by using the steepest descent method. To the best knowledge of the authors, the hybrid learning method has not been reported in type-2 FLS.

Only the BP learning method for type-2 FLS has been proposed in the literature, therefore one of the main contributions of this work is to implement a new hybrid learning algorithm for interval type-2 FLS, in view of the success of the hybrid learning method in type-1 FLS [2]. In [3, 4] it is shown that hybrid algorithms improve convergence over the BP method. In the forward pass, FLS output is calculated and the consequent parameters are estimated by either RLS [2] or REFIL [5] methods. In the backward pass, the error propagates backward, and the antecedent parameters are estimated by the BP method. In [3, 4] one of the proposed hybrid algorithms is based on RLS, since it is a benchmark algorithm for parameter estimation or systems identification. In addition, the parameter estimation method called REFIL, has also been used since it improves performance over RLS [5]. Convergence of the proposed methods has been practically tested; however mathematical proof is still to be done in general for hybrid learning algorithms.

This papers proposes a hybrid learning algorithm for interval type-2 FLS for antecedent and consequent parameter estimation during training process using input-output data pairs. In the forward pass, FLS output is calculated and the consequent parameters are estimated using REDCO [5] a recursive orthogonal least-square (OLS) learning method. In the backward pass, the error propagates backward, and the antecedent parameters are estimated by the BP method.

A second but very important purpose of this paper is to propose an application methodology based on interval type-2 FLS and the hybrid learning method mentioned above for hot strip mill (HSM) temperature prediction. Interval type-2 FLS is suitable for industrial modelling and control applications. The scale breaker (SB) entry mean and surface temperatures are used by the finishing mill set-up (FSU) model [6] to preset the finishing mill (FM) stand screws and to calculate the transfer bar thread speed, both required to achieve the FM exit target head gage the target head temperature.

In temperature prediction, the inputs of the fuzzy type-2 models, used to predict the SB entry temperatures, are the surface temperature of the transfer bar at the roughing mill (RM) exit $(x_1)$ and the time required by the transfer bar head to reach the SB entry zone $(x_2)$. Currently, the surface temperature is measured using a pyrometer located at the RM exit side. Scale grows at the transfer bar surface producing a noisy temperature measurement. The measurement is also affected by environment water steam as well as pyrometer location, calibration, resolution and repeatability. The head end transfer bar travelling time is estimated by the FSU model using FM estimated thread speed. Such estimation has an error associated with the inherent FSU model uncertainty. Although temperature prediction $(y)$ is a critical issue in a HSM the problem has not been fully addressed by fuzzy logic control systems [1, 3, 4].

The proposed algorithm is evaluated using an interval type-2 non-singleton type-2 FLS inference system (type-2 NSFLS-2) which predicts the transfer bar surface temperature at the SB entry zone.

This work is organized as follows. Section 2 gives the hybrid learning problem formulation for interval type-2 fuzzy logic systems. Section 3 presents solution as an adaptive training algorithm. Section 4 shows an interval type-2 NSFLS-2 application for HSM temperature prediction using the hybrid learning method. Conclusions are stated in Section 5.

## 2 Problem Formulation

Most of the industrial processes are highly uncertain, non-linear, time varying and non-stationary [3, 4, 7], having very complex mathematical representations. Interval type-2 FLS take easily the random and systematic components of type A or B standard uncertainty [8] of industrial measurements. The non-linearities are handled by FLS as identifiers and universal approximators of nonlinear dynamic systems [9, 10, 11]. The stationary noise and non-stationary additive noise are handled in natural way by interval type-2 FLS [1]. Such characteristics make interval type-2 FLS a very powerful inference system to model and control industrial processes

In [1] only one-pass and back-propagation (BP) algorithms are presented as interval type-2 FLS learning methods. Three basic problems for which it is not possible to use RLS on interval type-2 FLS are explained:

1. The starting point for the RLS method to designing an interval singleton FLS is a type-1 Fuzzy Basis Function (FBF) expansion. No such FBF expansion exists for a general type-2 non-singleton type-2 FLS. Since an interval type-2 FLS output $y(\mathbf{x})$ can be expressed as:

$$y(\mathbf{x}) = \frac{1}{2}\left[\mathbf{y}_l^T \mathbf{p}_l(\mathbf{x}) + \mathbf{y}_r^T \mathbf{p}_r(\mathbf{x})\right] . \quad (\mathbf{1})$$

   with $M$ ordered rules, it looks like a least-squares method can be used to tune the parameters in $\mathbf{y}_l^T$ (matrix transpose of M left-points $y_l^i$ of consequent centroids) and $\mathbf{y}_r^T$ (matrix transpose of M right-points $y_r^i$ of consequent centroids). Unfortunately, this is incorrect. The problem is that in order to know the FBF expansions $\mathbf{p}_l(\mathbf{x})$ and $\mathbf{p}_r(\mathbf{x})$, each $y_l^i$ and $y_r^i$ (the M left-points and right-points of interval consequent centroids) must be known first. Because at initial conditions of the calculations there are no numerical values for those elements, it is impossible to do this; hence the FBF expansions $\mathbf{p}_l(\mathbf{x})$ and $\mathbf{p}_r(\mathbf{x})$ cannot be calculated. This situation does not occur for type-1 FBF expansion.
2. Although $y_l$ and $y_r$ (the end-points of interval type-2 FLS center-of-sets type-reduced set $Y_{COS}$ ) can be expressed as an interval $\left[\underline{f}^i, \overline{f}^i\right]$ in terms of

their lower ( $\underline{f}^i$ ) and upper ( $\overline{f}^i$ ) M firing sets, and the corresponding M consequents left and right-points, $y_l^i$ and $y_r^i$, as:

$$y_l = y_l\left(\overline{f}^1,...,\overline{f}^L,\underline{f}^{L+1},...,\underline{f}^M,y_l^1,...,y_l^M\right). \quad (2)$$

$$y_r = y_r\left(\underline{f}^1,...,\underline{f}^R,\overline{f}^{R+1},...,\overline{f}^M,y_r^1,...,y_r^M\right). \quad (3)$$

where L and R are not known in advance [1]. L is the index to the rule-ordered FBF expansions at which $y_l$ is a minimum, and R is the index at which $y_r$ is a maximum. Once the points L and R are known, (1) is very useful to organize and describe the calculations of $y_l$ and $y_r$.

3. The next problem has to do with the re-ordering of $y_l^i$ and $y_r^i$ [1]. The type-1 FBF expansions have always had an inherent rule ordering associated with them; i.e. rules $R^1, R^2,..., R^M$ always established the first, second,…, and $Mth$ FBF. This order is lost and it is necessary to restore it for later use.

## 3 Problem Solution

### 3.1 Type-2 FLS

A type-2 fuzzy set, denoted by $\widetilde{A}$, is characterized by a type-2 membership function $\mu_{\widetilde{A}}(x,u)$, where $x \in X$ and $u \in J_x \subseteq [0,1]$ and $0 \le \mu_{\widetilde{A}}(x,u) \le 1.$:

$$\widetilde{A} = \left\{\left((x,u),\mu_{\widetilde{A}}(x,u)\right) \mid \forall x \in X, \forall u \in J_x \subseteq [0,1]\right\}. \quad (4)$$

This means that at a specific value of $x$, say $x'$, there is no longer a single value as for the type-1 membership function $(u')$; instead the type-2 membership function takes on a set of values named the primary membership of $x'$, $u \in J_x \subseteq [0,1]$. It is possible to assign an amplitude distribution to all of those points. This amplitude is named a secondary grade of general type-2 fuzzy set. When the values of secondary grade are the same and equal to 1, there is the case of an interval type-2 membership function [1, 12, 13, 14, 15].

### 3.2 Using Recursive OLS Learning Algorithm in Interval Type-2 FLS

Table 1 shows one pass learning algorithm activities BP method.

**Table 1.** One Pass In Learning Procedure for Interval Type-2 FLS

| | Forward Pass | Backward Pass |
|---|---|---|
| Antecedent Parameters | Fixed | BP |
| Consequent Parameters | Fixed | BP |

The proposed hybrid algorithm uses recursive OLS during forward pass for consequent parameters tuning and BP during backward pass for antecedent parameters tuning, as shown in Table 2.

**Table 2.** Two Passes In Hybrid Learning Procedure for Interval Type-2 FLS

| | Forward Pass | Backward Pass |
|---|---|---|
| Antecedent Parameters | Fixed | BP |
| Consequent Parameters | OLS | Fixed |

### 3.3 Adaptive OLS-BP Hybrid Learning Algorithm

The hybrid training method is based on the initial conditions of consequent parameters: $y_l^i$ and $y_r^i$. It presented as in [1]: Given N input-output training data pairs, the hybrid training algorithm for E training epochs, should minimize the error function

$$e^{(t)} = \frac{1}{2}\left[f_{s2}\left(\mathbf{x}^{(t)}\right) - y^{(t)}\right]^2 \quad . \tag{5}$$

## 4 Application to Transfer Bar Surface Temperature Prediction

### 4.1 Hot Strip Mill

Because of the complexities and uncertainties involved in rolling operations, the development of mathematical theories has been largely restricted to two-dimensional models applicable to heat losing in flat rolling operations.

Fig. 1, shows a simplified diagram of a HSM, from the initial point of the process at the reheat furnace entry to its end at the coilers.

Besides the mechanical, electrical and electronic equipment, a big potential for ensuring good quality lies in the automation systems and the used control techniques. The most critical process in the HSM occurs in the FM. There are several mathematical model based systems for setting up the FM. There is a model-based set-up system [6] that calculates the FM working references needed to obtain gauge, width and temperature at the FM exit stands. It takes as inputs: FM exit target gage, target width and target temperature, steel grade, hardness ratio from slab chemistry, load distribution, gauge offset, temperature offset, roll diameters, load distribution, transfer bar gauge, transfer bar width and transfer bar temperature entry.

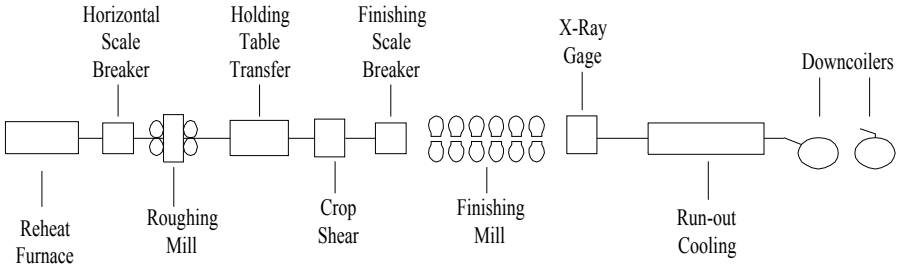



**Fig. 1.** Typical hot strip mill

The errors in the gauge of the transfer bar are absorbed in the first two FM stands and therefore have a little effect on the target exit gauge. It is very important for the model to know the FM entry temperature accurately. A temperature error will propagate through the entire FM.

### 4.2 Interval Type-2 Fuzzy Logic System Design

The architecture of the FLS was established in such way that parameters are continuously optimized. The number of rule-antecedents was fixed to two; one for the RM exit surface temperature and the other for transfer bar head traveling time. Each antecedent-input space was divided in five fuzzy sets, fixing the number of rules to twenty five. Gaussian primary membership functions with uncertain means were chosen for both, the antecedents and consequents. Each of the rules of the interval type-2 NSFLS-2 is characterized by six antecedent membership function parameters and two consequent parameters. Each input value has two standard deviation parameters: given ten parameters per rule.

The resulting interval type-2 FLS uses type-2 non-singleton fuzzification, maximum t-conorm, product t-norm, product implication and center-of-sets type-reduction.

### 4.3 Noisy Input-Output Training Data Pairs

From an industrial HSM, noisy input-output pairs of three different coil types were collected and used as training and checking data. The inputs were the noisy measured RM exit surface temperature and the measured RM exit to SB entry transfer bar traveling time. The output was the noisy measured SB entry surface temperature.

### 4.4 Input Membership Function

The primary membership functions for each input of the interval type-2 NSFLS-2 was:

$$\mu_{X_k}(x_k) = \exp\left[-\frac{1}{2}\left[\frac{x_k - x_k^{'}}{\sigma_{X_k}}\right]^2\right]. \quad (6)$$

where: $k = 1,2$ (the number of type-2 non-singleton inputs), $\mu_{X_k}(x_k)$ is centered at $x_k = x_k^{'}$ and $\sigma_{X_k}$ is the standard deviation whose values varies over an interval of values $[\sigma_{k1}, \sigma_{k2}]$. The standard deviation of the RM exit surface temperature measurement, $\sigma_{X_1}$, initially varies over [11.0, 14.0] $^{o}C$ interval, whereas the standard deviation head end traveling time measurement, $\sigma_{X_2}$, initially varies over [1.41, 3.41] s interval. The uncertainty of the input data was modeled as non-stationary additive noise using type-2 fuzzy sets.

### 4.5 Antecedent Membership Functions

The primary membership function for each antecedent was a Gaussian with uncertain means as:

$$\mu_k^i(x_k) = \exp\left[-\frac{1}{2}\left[\frac{x_k - m_k^i}{\sigma_k^i}\right]^2\right]. \quad (7)$$

where $m_k^i \in \left[m_{k1}^i, m_{k2}^i\right]$ is the uncertain mean, $\sigma_k^i$ is the standard deviation, $k = 1,2$ (the number of antecedents) and $i = 1,2,..25$ (the number of M rules). The means of the antecedent fuzzy sets were uniformly distributed over the entire input space. $m_{11}$ and $m_{12}$ are the upper and lower values of the uncertain mean, and $\sigma_1$ is standard deviation of input $(x_1)$. $m_{22}$ and $m_{22}$ are the upper and lower values of the uncertain mean and $\sigma_2$ is standard deviation of input $(x_2)$.

### 4.6 Fuzzy Rule Base

The type-2 fuzzy rule base consists of a set of IF-THEN rules that represents the model of the system. The interval non-singleton type-2 FLS have two inputs $x_1 \in X_1, and, x_2 \in X_2$ and one output $y \in Y$, which have a corresponding rule base size of $M$ = 25 rules of the form:

$$R^i : IF \quad x_1 \quad is \quad \tilde{F}_1^i \quad and \quad x_2 \quad is \quad \tilde{F}_2^i, \quad THEN \quad y \quad is \quad \tilde{G}^i \ . \tag{8}$$

where $i$ = 1,2,…25, $\tilde{F}_1^i$ is the $(x_1)$ input type-2 fuzzy set, $\tilde{F}_2^i$ is $(x_2)$ input type-2 fuzzy set and $\tilde{G}^i$ is the consequent type-2 fuzzy set. These rules represent a fuzzy relation between the input space $X_1 \times X_2$ and the output space $Y$, and it is complete, consistent and continuous [16].

### 4.7 Consequent Membership Functions

The primary membership function for each consequent is a Gaussian with uncertain means, as defined in (7). Because the center-of-sets type-reducer replaces each consequent set $\tilde{G}^i$ by its centroid, then $y_l^i$ and $y_r^i$ are the consequent parameters.

Because only the input-output data training pairs $\left(x^{(1)} : y^{(1)}\right), \left(x^{(2)} : y^{(2)}\right),\ldots,$ $\left(x^{(N)} : y^{(N)}\right)$ are available and there is no data information about the consequents, the initial values for the centroid parameters $y_l^i$ and $y_r^i$ may be determined according to the linguistic rules from human experts or be chosen arbitrarily in the output space [16]. In this work the initial values of parameters $y_l^i$ and $y_r^i$ are such that the corresponding membership functions uniformly cover the output space.

### 4.8 Results

An interval type-2 NSFLS-2 system was used to predict the transfer temperature. For each of the two methods, BP and hybrid OLS-BP, we ran fifteen epoch computations; using eighty-seven input-output training data pairs, 250 parameters were tuned. The performance evaluation for the learning methods was based on the benchmarking root mean-squared error (RMSE) criteria [1]:

$$RMSE_{s2}(*) = \sqrt{\frac{1}{n}\sum_{k=1}^{n}\left[Y(k) - f_{s2-*}\left(\mathbf{x}^{(k)}\right)\right]^2} \ . \tag{9}$$

where $Y(k)$ is the output training data from the model using ten check data pairs, $RMSE_{s2}(*)$ stands for $RMSE_{s2}(BP)$, and for $RMSE_{s2}(OLS-BP)$, and were obtained when applied BP and hybrid OLS-BP learning methods to an interval type-2

NSFLS-2. Fig. 2, shows RMSE of the two used interval type-2 NSFLS-2 with fifteen epochs' computations for the case of type A coils. It can be appreciated that after four epochs, the hybrid OLS-BP has better performance than BP method.

## 5 Conclusion

In this paper we have developed an orthogonal-BP hybrid algorithm to train an interval type-2 NSFLS-2 and used to predict HSM transfer bar temperature. The interval type-2 NSFLS-2 antecedent membership functions and consequent centroids successfully absorbed the uncertainty introduced by the training noisy data. The uncertainty of the input data measurements was modeled as stationary additive noise using type-2 fuzzy sets. The selected initial values of the antecedent and consequent parameters can affect the results of the interval type-2 FLS predictions. BP and OLS-BP methods were tested and parameters estimation has been demonstrated. There is a substantial improvement in performance and stability of the hybrid method over the only BP method. The hybrid OLS-BP achieves the better RMSE performance as can be seen in the experimental results. It has been shown that the proposed methodology can be applied in modeling and control of the steel coil temperature. It has also been envisaged its application in gage, width and flatness prediction.

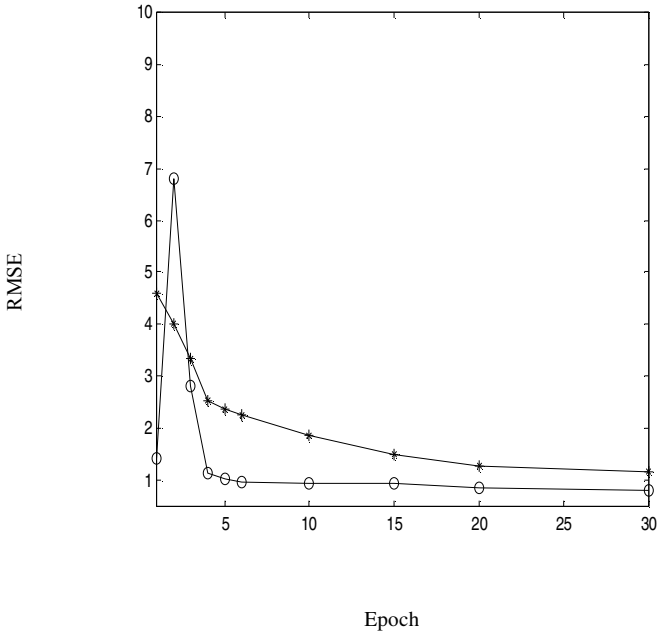



**Fig. 2.** Type-2 NSFLS-2 (*) RMSEs2 (BP) (o) RMSEs2 (OLS-BP)

## References

1. Mendel, J. M. : Uncertain Rule Based Fuzzy Logic Systems: Introduction and New Directions, Upper Saddle River, NJ, Prentice-Hall, (2001)
2. Jang, J. -S. R., Sun, C. -T., Mizutani, E.: Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Upper Saddle River, NJ: Prentice-Hall, (1997)
3. Mendez, M., Cavazos, A., Leduc, L. , Soto, R.: Hot Strip Mill Temperature Prediction Using Hybrid Learning Interval Singleton Type-2 FLS, Proceedings of the IASTED International Conference on Modeling and Simulation, Palm Springs, February (2003), pp. 380-385
4. Mendez, M., Cavazos, A., Leduc, L. , Soto, R.: Modeling of a Hot Strip Mill Temperature Using Hybrid Learning for Interval Type-1 and Type-2 Non-Singleton Type-2 FLS, Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, Benalmádena, Spain, September (2003), pp. 529-533
5. Aguado, A.: Temas de Identificación y Control Adaptable, La Habana 10200 Cuba, Instituto de Cibernética, Matemáticas y Física, (2000)
6. GE Models, Users reference, Vol. 1, Roanoke VA, (1993)
7. Lee, D. Y., Cho, H. S.: Neural Network Approach to the Control of the Plate Width in Hot Plate Mills, International Joint Conference on Neural Networks, (1999), Vol. 5, pp. 3391-3396
8. Taylor, B. N., Kuyatt, C. E.: Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results, September (1994), NIST Technical Note 1297
9. Wang, L-X.: Fuzzy Systems are Universal Approximators, Proceedings of the IEEE Conf. On Fuzzy Systems, San Diego, (1992), pp. 1163-1170
10. Wang, L-X., Mendel, J. M.: Back-Propagation Fuzzy Systems as Nonlinear Dynamic System Identifiers, Proceedings of the IEEE Conf. On Fuzzy Systems, San Diego, CA. March (1992), pp. 1409-1418
11. Jang, J. -S. R., Sun, C. -T.: Neuro-Fuzzy Modeling and Control, The Proceedings of the IEEEE, Vol. 3, Mach (1995), pp. 378-406
12. Liang, Q. J., Mendel, J. M.: Interval type-2 fuzzy logic systems: Theory and design, Trans. Fuzzy Sist., Vol. 8, Oct. (2000), pp. 535-550
13. John, R.I.: Embedded Interval Valued Type-2 Fuzzy Sets, IEEE Trans. Fuzzy Sist., (2002)
14. Mendel, J. M., John, R.I.: Type-2 Fuzzy Sets Made Simple, IEEE Transactions on Fuzzy Systems, Vol. 10, April (2002)
15. Mendel, J.M.: On the importance of interval sets in type-2 fuzzy logic systems, Proceedings of Joint 9th IFSA World Congress and 20th NAFIPS International Conference, (2001)
16. Wang, L-X.: A Course in Fuzzy Systems and Control, Upper Saddle River, NJ: Prentice Hall PTR, (1997)

# Characterization and Interpretation of Classes Based on Fuzzy Rules in ill-Structured Domains

Fernando Vázquez[1], Juan Luis Díaz de León[2]

[1] UPIICSA-IPN, Computer Science Dept., México, D.F.
fvazquez_t@hotmail.com
[2] CIC-IPN, Computer Science Dept., México, D.F.
jdiaz@cic.ipn.mx

**Abstract.**
Nowadays, when a classification is given from a set of objects, it seems to be necessary to make use of tools to assist users to interpret tasks, in order to establish the semantic structure of the resultant classes from a given classification. It is often enough for the user to build the classes automatically, but he needs a sort of tool to help himself to understand the reason why such classes were detected there. CIADEC is a computer system that implements the methodology AUGERISD, which allows us to obtain the automatic characterization and interpretation of conceptual descriptions, combining: concepts, artificial intelligence techniques, inductive learning and statistics. A system based on fuzzy rules to find out a characterization of the given classes by an automatic form is described in this paper. A specific case applied to Wastewater Treatment Plant (WWTP) shows the stages for this methodology.

## 1 Introduction

The aim of this paper is to present a methodology which combines statistical tools through inductive learning, in such a way that it is the base of statistics analysis for several (numeric) measurements. Such methodology can identify the characteristic situations (classes) which can be found in the plant and it also produces a *conceptual description* of them. Once they are identified and *understood* by the user, this typical situations may be used afterwards to support the process of the decision making. This decision making may be either automatic or not.

The process that was used helps us to know that the outflow water quality (according to quality water standards) is really complex. On the other hand, due to intrinsic features of wastewater and the consequences of a bad administration of the plant, such process is complex and delicate [3].

Just to have a general idea of what is happening in the plant, we provide a very brief description of the process: the water flows sequentially through three or four stages which are commonly known as pretreatment, primary, secondary and advanced treatment (see [4] for a detailed description of the process). Figure 1 depicts its general structure.

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 261-270*

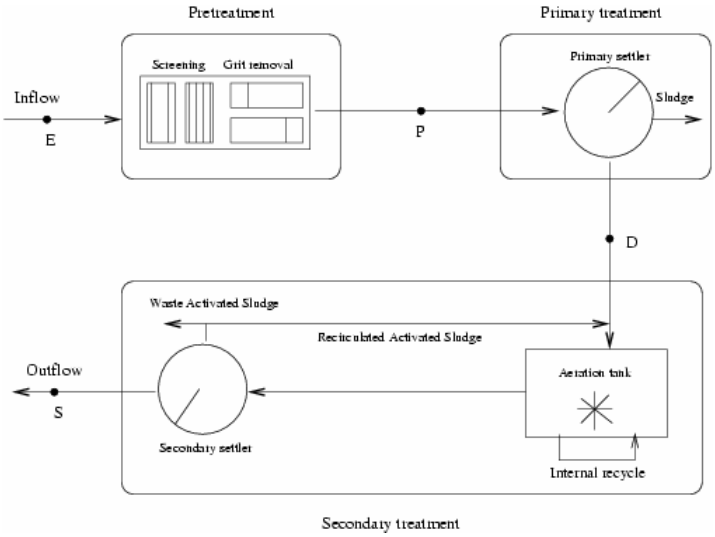



**Fig. 1.** General structure of wastewater treatment plant.

The paper is organized as follows: In section 2 the presentation of the Wastewater Treatment Plant (WWTP) data is shown; in section 3 the methodology for the characterization of the classes is described; in section 4 the methodology is applied to a WWTP; in section 5 the interpretation of the results obtained of the automatic characterization and interpretation of conceptual descriptions taken as references from the partition of Klass+ from which four classes are displayed. Finally, in section 6 the conclusions and future research are presented.

## 2 Presentation of the Wastewater Treatment Plant (WWTP) data

The set of qualitative and quantitative data analyzed in this research was taken from a treatment plant on the Catalonian coast (Lloret city, Spain). It was made up of 218 observations taken from the same number of consecutive days. Some of the observations were taken at different stages of the process, whereas others were taken based on calculations from those. The first ones correspond to the daily average of repeated measurements on a total of 63 attributes that were measured at various points in the facility (AB: the entrance of the center, SP1: the outlet of the first settling tank, B: in the biological reactor, SP3: in the third settling tank, AT: the treated water), as well as a description of the plant's state at the moment of measurement.

## 3 Methodology for the characterization of the classes

The objective is to generate a fuzzy rules system from a set of data which has been described by the attributes above mentioned; and to obtain a description that can be easily understandable for the user, and which indicates particularities from each class among the rest of them in a given partition $P = \{c_1,.....,c_\xi\}$.

*3.1 Statistical description of the attributes*

In this first stage, some classical descriptive techniques are used in order to identify the behavior and nature of the data referring to the data matrix $X$ . This stage is useful to obtain preliminary information about the variability of the measurements and to represent a multiple box-plot which will let us observe the relation between the attributes and the classes, and it is especially useful to represent the difference among clusters or classes.

*3.2 The use of multiple box-plots as a graphic tool for the detection of characterizing attributes.*

As we have already pointed out, the *multiple box-plots* are excellent base for this research, as a tool to view and compare the distribution of an attribute through all the classes. In this representation, it is possible to identify the characterizing attributes from class $C$ , explained through the concept of proper value from a class $C$ . It is quite simple to graphically observe if the multiple box-plot of a certain class doesn't intersect others; in such a case, the attribute is fully characterizing[1]. Sometimes, it is only a part of the box-plot that is not intersected; in this case it means that we have a partial characterizing variable.

*3.3 Study of classes interactions*

In this process, it is utterly important to consider the attributes, in their natural state, avoiding any arbitrary transformation about their nature that could change the sense of the intersection. This stage consists on identifying all the intersections occurring among the values of attributes and the different classes. We determine at what point in the range of attributes these intersections are changing, thus allowing identification of the different combinations existing among classes where the same value of a certain variable or attribute can be found and, consequently, to let proper values emerge (characterizing values). This would tell us whether they were fully or partially characterized.

*3.4 Space discreteness of attributes*

Exact intersections can be found with minimal computational cost, by simply calculating the minimum and maximum values by variable and class, and ordering

[1] Onwards, we will simple call characterizing variable of that class.

them. From this order, the discreteness of the variable is defined by a set of intervals of variable length $I^k = \{I_1^k + I_2^k, ....., I_{2\xi-1}^k\}$ so that $U_{s=1}^{2\xi-1} I_s^k = D_k$, in which the *unique values* of a variable in all the classes can be identified.

To extend these concepts, thus, if $m_C^k$ and $M_C^k$ are the minimum and maximum of the variable $X_k$ in the class $C \in P$, which have been observed in the descriptive of the multiple box-plots, where $m_C^k = \min_{i \in C}\{x_{ik}\}$ and $M_C^k \max_{i \in C}\{x_{ik}\}$. Now, we order them in ascendant form, and this is:

- Define $M^k$ like the set of all the minimums and maximums corresponding to the variable $X_k$, in all $P$ classes, this is:
  $M^k = \{m_{c1}^k, ...., m_{c\xi}^k, M_{c1}^k, ....M_{c\xi}^k\}$, being the $Card(M^k) = 2\xi$
- Ordering $M^k$ from minimum to maximum, a set is constructed $Z^k$ in the following way: $Z^k = \{z_i^k; i = 1, ...2\xi\}$, so:
  i. $z_1^k = \min M^k$ and
  ii. $z_i^k = \min(M^k \setminus \{z_j^k; j < i\}), i = 2, ...., 2\xi$

  Since $Z^k = \{z_i^k\}$ is an ordered set, its elements have the following property: $Z^K = \{z_J^K \mid z_{J-1}^K < z_j^k : 1 < j < 2\xi\}$, this set is called *cut points*.
- From this ordered set, we build the intervals system of variable length $I^k$ in the following form: $I^k = \{I_s^k : 1 \le s \le 2\xi - 1\}$, in which:
  i. $I_1^k = [\, z_1^k, z_2^k\,]$
  ii. $I_s^k = (\, z_s^k, z_{s+1}^k]$, with $s = 2, ..., 2\xi - 1$

A new categorizing variable is then defined as $I^k$; whose set of values is $D^k = \{I_1^k, ...., I_{2\xi-1}^k\}$. $I^k$ identifies all intersections among classes that $X_k$ defines, representing a system of length intervals associated to such variable. Thus, were $2\xi$ different cut points, then $2\xi - 1$ intervals the most are generated and $Card(D^k) = 2\xi - 1$, taking into account that $\xi$ is the number of initial classes of reference that we want to characterize.

On the other hand, since $D^k$ is the domain of $X_k$, $D^k$ represents the categorization of itself, yet not arbitrary at all, it is also calculated immediately. Finally, it is

necessary to observe that in order to construct $I^k$ it is not necessary to perform the multiple box-plots anymore, even though it is still an excellent representation of what it is being done.

*3.5 Construction of distribution table of classes versus intervals*

From a system of intervals, a contingency table is set for a variable $X_k$, as a matrix A set of numbers in which each line represents an interval $I^k$ and each column represents a class found in the previous stage, for the classes reference partition P. Therefore, any given cell in this matrix indicates the number of elements in the domain $I$, whose values of $X_k$ are found in an interval represented by $I_s^k$.

In general, for a given value of the variable $X_k$, objects from different classes are found.

*3.6 Generating a fuzzy rules system* $R(X_k, P)$

From this distribution from table *B*, a rule system is constructed for each non-null cell $p_{sc}$. From this matrix, the following rules are generated: if $X_{ik} \in I_s^k \xrightarrow{psc} C$.

This way, $R(X_k, P)$ can be used to recognize the class (or classes) belonging to a certain day in which $i = (x_{i1}, ....., x_{iK}, ...., x_{ik})$ belongs to, according to its value of $X_k$.

*3.7 Interpretation of resultant classes*

The interpretation of the resultant classes tends to be utterly important to use the generated knowledge as an aiming tool for the future decision taking. In fact, it has been remarked that the validation of a classification has been considered as the degree of interpretability and/or the utility of these, without any other criteria but that of a specialist that looks at the resultant classes.

Having the conditioned table of distributions as a base to get the previously described intervals in 3.4 from this section, any spare part *I* can be associated to its degree of belonging to each class. This info gives us a graphic of diffused degrees of belonging for each class and for each variable shown in figure 3. In the graphic, the horizontal axe is common and represents the range $X_k$. For every single class the degree of belonging of values of k is represented according to the rules. The scaffolding shape for such functions of belonging must be categorized from $X_k$ in $I^k$. Thus, given a value from $X_k$, it is easily noticeable its relation to other classes.

## 4 Application of the methodology to WWTP data

In this section, the methodology AUGERISD [5] is applied to the data that were gathered from the Wastewater Treatment Plant with Klass$^{+}$ [1] being partitioned into four classes. We applied the methodology automated through system CIADEC [6] to identify the relevant characteristics of the reference partition, having a fuzzy rules system obtained that will allow us to get the characterization and interpretation of the conceptual descriptions for the resulting classes of such reference partition, considering the analysis of all the attributes in such a way.

## 5 Generation of Interpretations

In [1] and [2] a first approach for an efficient algorithm which generates automatic interpretations of the classes is introduced. In this research, the main work is about the interpretation of the classes on the basis of categorical attributes. In this section, we are interested in the use of numerical attributes for interpreting classes. The real application we presented here is especially indicated for this goal, since categorical attributes are not presented in the data matrix at all and only numerical measurements are useful to describe data.

From this automated methodology we obtain the conditioned distributions to each interval (see Table 1). The table, which has been mentioned above, gives us the percentage of elements of certain interval in each class, obtaining a reduced fuzzy rules system (see figure 2).

**Table 1. Conditional Distribution Table of Attribute $Q - AB$**

| Intervals | Classes | | | |
|---|---|---|---|---|
| | C1 | C2 | C3 | C4 |
| [4910, 5881] | 0.94 | 0.00 | 0.06 | 0.00 |
| (5881, 6277] | 0.85 | 0.07 | 0.08 | 0.00 |
| (6277, 13454] | 0.41 | 0.23 | 0.36 | 0.00 |
| (13454, 13563] | 0.00 | 0.34 | 0.33 | 0.33 |
| (13563, 13563] | 0.00 | 0.00 | 0.00 | 0.00 |
| (13563, 14375] | 0.00 | 0.50 | 0.50 | 0.00 |
| (14375, 23394] | 0.00 | 1.0 | 0.00 | 0.00 |

Having any of them, since it is based on the conditioned distributions table to the intervals, they can be associated with an object (day) that is called: a belonging degree for each class. This idea gives place to a graph of a belonging fuzzy degree for each class and for each attribute (see figure 3).

$$r_1 : x_{Q-AB,i} \in [4910 \ , 5881 \ ] \xrightarrow{0.94} C1$$
$$r_2 : x_{Q-AB,i} \in (5881 \ , 6277 \ ] \xrightarrow{0.85} C1$$
$$r_3 : x_{Q-AB,i} \in (6277 \ , 13454 \ ] \xrightarrow{0.41} C2$$
$$r_4 : x_{Q-AB,i} \in (13454 \ , 13563 \ ] \xrightarrow{0.34} C2$$
$$r_6 : x_{Q-AB,i} \in (13563 \ , 14375 \ ] \xrightarrow{0.5} C2$$
$$r_7 : x_{Q-AB,i} \in (14375 \ , 23394 \ ] \xrightarrow{1.0} C2$$

**Fig. 2. Fuzzy Rules Reduced System of Attribute** $Q-AB$

In this way, from a system with a method of creation of linguistic labels we automatically generate conceptual descriptions for these classes.

Using this methodology, the following interpretations are produced. The opinion of the experts is also included in the discussion below, as they confirmed the understandability of the discovered classes.

Class 1:

The output water is clean. The input water is low dirty (values are low in almost all the attributes). Experts identified this class with the class of those days with very good plant performance, as a consequence of the good conditions, even ammonium is reduced.

Class 2:

High wastewater inflow. The water which comes in is medium dirty (intermediate values in almost all the attributes: suspended solid total, chemical organic matter, biodegradable organic matter, etc). Settler is making high effect (levels of suspended and volatile suspended solid are significantly reduced at the primary treatment). From the expert point of view, this class is interpreted as the class of days with general good performance, but in which punctual problems can produce some isolated parameter with high values.

Class 3:

The water that comes in is very dirty. The output water with intermediate measures all the attributes. Nonetheless, the performance of the plant is not so good. This class contains some days in which isolated control parameters overcome the permitted values.

Experts identified this class to the class of those days with organic material overloading on summer days with optimal WWTP-operation.

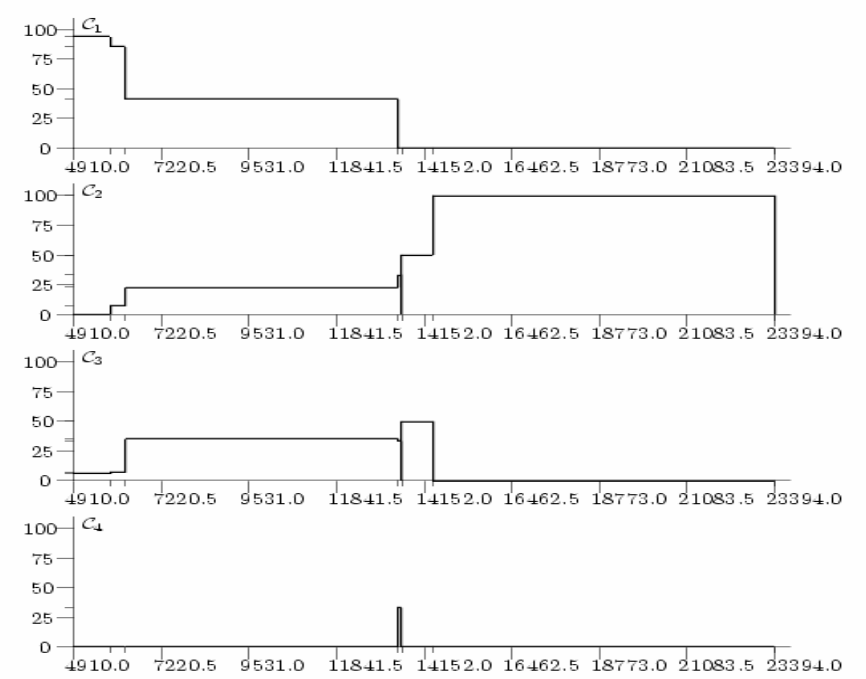



**Fig. 3. Attribute Graphic** $Q - AB$

*Class 4:*

High levels of chlorine and conductivity. From the expert point of view, this class is interpreted as the class of days with chlorine over amount.

## 6 Conclusions and Future Work

This work constitutes a positive experience in terms of establishing a formal methodology to automatically obtain conceptual interpretations of the classes, on the basis of numerical attributes used to describe objects (days in this case).

From a very small and partial set of rules, the system created a new level of abstraction that catches the nature of a Wastewater Treatment Plant for the given set of data (noisy, incomplete and heterogeneous), producing a set of identified classes, as well as their conceptual interpretation, which was directly interpreted by the experts on this matter.

In the future, it is our intention to explore the series of time of the behavior of the plots in their variables.

**Acknowledgements**
This research has been partially financed by COFAA and IPN, México.

## References

1. Gibert K. In L'us de la informació simbólica en *la automatització del tractament estadístic de dominis poc estructurats*. Ph D. Thesis, UPC, BCNA, 1994.
2. Gibert K., Aluja T. and Cortés. *Knowledge discovery with clustering based on rules, interpreting results*
3. Gimeno J.M., Béjar I., Sànchez-Marrè and Cortés U. In *"Discovering and modelling process change: An application to industrial processes"*. Practical Applications of Data Mining and Knowledge Discovery. 1997.4. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
4. Metcalf and Eddy Inc., In *Wastewater engineering: treatment/disposal/reuse*. McGraw-Hill, 1991.
5. Vázquez F., Gibert K. In *Automatic Generation of Fuzzy Rules in ill-Structured Domanis with Numerical Variables*, publisher: UPC, LSI, Report num: LSI-01-51-R. Barcelona, España. 2001. E-mail: http://www.lsi.upc.es.
6. Vázquez F., Gibert K. In Implementation of the methodology "Automatic Characterization and Interpretation of Conceptual Descriptions in ill-Structured Domains using Numerical Variables", publisher: UPC, LSI, Report num: LSI-02-28-R. Barcelona, Spain. 2002. E-mail: http://www.lsi.upc.es.
7. Rodas J., Alvarado G. and Vázquez F. Using the KDSM methodology for knowledge discovery from a labor domain. (SNPD2005). Towson University. Towson, Maryland, USA. May 2005.

# Differential Evolution Algorithms to Solve Optimal Control Problems Efficiently

I.L. López-Cruz; A. Rojano-Aguilar
Postgrado en Ingeniería Agrícola y Uso Integral del Agua
Universidad Autónoma Chapingo
Chapingo, México
e-mail:ilopez@correo.chapingo.mx

**Abstract.** Optimal control of multimodal and singular problems of bioreactors has received considerable attention recently. Three main approaches have been attempted: deterministic methods like Iterative Dynamic Programming, stochastic methods like Adaptive Stochastic algorithms, and Evolutionary Algorithms. The aim of this research is to demonstrate that new evolutionary algorithms called generically Differential Evolution (DE) algorithms are efficient in solving both multimodal, and singular optimal control problems especially when a relatively greater number of variables (50-100) have to be optimized. DE algorithms are simple and efficient evolutionary methods when are compared to other evolutionary methods regarding the number of function evaluations to converge to a solution. It is shown that besides he three main operators of DE: mutation, crossover and selection, a filter operator is added in order to obtain smoother optimal trajectories of singular optimal control problems.

## 1 Introduction

During the last decade interest on the application of global optimization methods in optimal control has significantly increased. Evolutionary Algorithms are stochastic optimization methods that have shown several advantages as global optimization methods. They have been applied in the past basically to solve static optimization problems and only rarely to solve multimodal optimal control problems. It is well known that optimal control problems with singular arcs are very hard to solve by using the Pontryagin minimum principle [1], [2]. Singular optimal control problems are frequently found in the optimization of bioreactors [3], [4] and likely also in other biosystems [5]. Also multimodal optimal control problems are frequently found in optimization of bioreactors [6]. Luus [6,7] has applied Iterative Dynamic Programming (IDP), which can be considered as another global optimization method, to solve multimodal and also singular control problems. Tholudur and Ramirez [8], who also used IDP, found highly oscillatory behavior of optimal control trajectories in solving singular optimal control problems. Therefore, they proposed two filters in order to calculate smoother optimal trajectories. Recently, Roubos *et al*. [5] suggested two smoother evolutionary operators for a Genetic Algorithm with floating-point representation of the individuals and applied this approach to calculate solutions for two fed-batch bioreactors.

In spite of its reliability as a global optimization method, IDP is rather complex with several algorithm parameters, which require an expensive tuning, before the application of the algorithm to a new problem. Since many experiments are necessary IDP becomes deceptively inefficient recalling that the computation time is critical in solv-

*© A. Gelbukh, R. Monroy. (Eds.)*
*Advances in Artificial Intelligence Theory*
*Research on Computing Science 16, 2005, pp. 271-280*

ing optimal control problems. In dynamic optimization each evaluation of the cost function means running a long simulation (integration) of the dynamic model of the process. Theoretical and empirical results [9] have shown that Evolutionary Algorithms (like those based in Genetic Algorithms) that use low mutation rates for mutation and high probability for crossover are not good candidates to solve optimal control problems efficiently since they may require highly number of function evaluations when many variables are optimized or these variables are correlated. Therefore, there is a necessity of developing more efficient global optimization algorithms for solving optimal control problems, in general, and multimodal and singular optimal control problems, in particular.

Lately, a new family of evolutionary algorithms named Differential Evolution (DE) has been proposed [10, 11] which is not only simple but also remarkably efficient compared to other Evolutionary Algorithms, Simulated Annealing and Stochastic Differential equations. Recently, results were have presented that show DE are one of the most efficient evolutionary algorithms to solve optimal control problems efficiently [12, 13]. The present work illustrates that indeed DE algorithms are good candidates to solve multimodal optimal control problems. Also modified DE algorithms are evaluated in solving singular optimal control problems. The new operator is simple and does not add any additional algorithm parameter. The so-called median filter operator basically consists of a sliding window such as each control is replaced with the median of a few neighboring controls. The proposed operator is implemented on the *DE/rand/bin/1* algorithm, and tested on solving a dynamic optimization problem of a fed-batch bioreactor. In this work efficiency of algorithms is measured by counting the number of function evaluations required to solve a problem, which is a machine independent criterion. A comparison of the *DE/rand/bin/1* algorithm performance with and without the smoother operator is presented to illustrate the advantages of the proposed modified Differential Evolution algorithm.

## 2 The Optimal Control Problem

A continuous-time optimal control problem [12] implies to find an optimal control $u^*(t)$, which causes the system

$$\dot{x} = f(x((t),u(t),t)\ ,\ x(t_0) = x_0 \qquad (1)$$

to follow an admissible trajectory $x^*(t)$ that optimizes the performance measure given by the functional :

$$J = \phi(x(t_f),t_f) + \int_0^{tf} L(x(t),u(t),t)dt\ . \qquad (2)$$

where $x \in R^n$ denotes the states of the system and $u \in R^m$ denotes a control vector. In addition the controls are constrained $\alpha \le u(t) \le \beta$. The final time $t_f$ is fixed. As the Hamiltonian function:

$$H(t) = \lambda^T(t) f(x(t),u(t),t)\ . \qquad (3)$$

is linear with respect to the controls, the optimal control problem becomes singular [13]. Singular optimal control problems are difficult to solve by classical methods and direct methods seem to be a promising approach. To apply a direct optimization method a parameterization of the controls is necessary, for instance piecewise constant control can be applied

$$u(t) = u(t_k),\ t \in [t_k, t_{k+1}),\ k = 0,1,...N-1 \quad (\mathbf{4})$$

where N is the number of sub-intervals for the time interval $[t_0, t_f]$. In this way a vector of parameters $\widetilde{u} = [u_1^T, u_2^T, ..., u_{N-1}^T]$ is defined and the value that optimizes the original performance index (2) can be obtained by parameter optimization methods or solving a Non-Linear Programming (NLP) optimization problem. The numerical solution of these problems is challenging due to the non-linear and discontinuous dynamics. Likely, there is not a unique global solution. Standard gradient-based algorithms are basically local search methods; they will converge to a local solution. In order to surmount these difficulties global optimization methods must be used in order to ensure proper convergence to the global optimum.

## 3 Differential Evolution Algorithms

A differential evolution algorithm is as follows:

```
Generate a population ( P(0) ) of solutions.
Evaluate each solution.
g=1;
while (convergence is not reached)
     for i=1 to μ
      Apply differential mutation.
      Execute differential crossover.
      Clip the new solution if necessary.
      Evaluate the new solution.
      Apply differential selection.
     end
     g=g+1;
end
```

Firstly, a population $P(0)$ of floating-point vectors $\bar{u}_i, i = 1,...,\mu$ is generated randomly from the domain of the variables to be optimized, where $\bar{u} = [u_1,...,u_d]$ and $\mu$ denotes the population size. Next, each vector is evaluated by calculating its associated cost function (eqn. 2), $i = 1,...,\mu$. Notice that the evaluation of each solution implies to carry out a numerical integration of the dynamic model (1). After that, a loop begins in which the evolutionary operators: differential mutation, differential crossover and selection are applied to the population ( $P(g)$ ), where $g$ denotes a generation number. Differential Evolution operators are quite different than those frequently found in other evolutionary algorithms. In DE, the differential mutation operator consists of the generation of $\mu$ mutated vectors according to the equation:

$$\vec{v}_i = \vec{u}_{r_1} + F \cdot (\vec{u}_{r_2} - \vec{u}_{r_3}),\ i = 1,2,...,\mu\ . \quad (5)$$

where the random indices $r_1, r_2, r_3 \in [1,2,...,\mu]$ are mutually different and also different from the index *i*. $F \in [0,2]$ is a real constant parameter that affects the differential variation between two vectors. Greater values of $F$ and/or the population size ($\mu$) tend to increase the global search capabilities of the algorithm because more areas of the search space are explored.

The crossover operator combines the previously mutated vector $\vec{v}_i = [v_{1i}, v_{2i}, ..., v_{di}]$ with a so-called target vector (a parent solution from the old population) $\vec{u}_i = [u_{1i}, u_{2i}, ..., u_{di}]$ to generate a so-called trial vector $\vec{u}'_i = [u'_{1i}, u'_{2i}, ..., u'_{di}]$ according to:

$$u'_{ji} = \begin{cases} v_{ji} & if\ \ (randb(j) \le CR)\ \ or\ \ \ j = rnbr(i) \\ u_{ji} & if\ \ (randb(j) > CR)\ \ and\ \ j \ne rnbr(i) \end{cases};\ j = 1,2,...,d;\ i = 1,2,...,\mu \quad (6)$$

where $randb(j) \in [0,1]$ is the j-th evaluation of a uniform random number generator, $rnbr(i) \in 1,2,...,d$ is a randomly chosen index. $CR \in [0,1]$ is the crossover constant, a parameter that increases the diversity of the individuals in the population. Greater values of CR give rise to a child vector ($\vec{u}'_i$) more similar to the mutated vector ($\vec{v}_i$). Therefore, the speed of convergence of the algorithm is increased. As can be seen from equation (6), each member of the population plays once the role of a target vector. It is important to realize that even when $CR = 0$, equation (6) ensures that parent and child vectors differ by at least one gene (variable). The three algorithm parameters that steer the search of the algorithm, the population size ($\mu$), the crossover constant ($CR$) and differential variation factor ($F$) remain constant during an optimization.

The selection operator compares the cost function value of the target vector $\vec{u}_i$ with that of the associated trial vector $\vec{u}'_i$, $i = 1,2,...,\mu$ and the best vector of these two becomes a member of the population for the next generation. That is,

$$\text{if } \phi(\vec{u}'_i(g)) < \phi(\vec{u}_i(g)) \text{ then } \vec{u}_i(g+1) := \vec{u}'_i(g)$$
$$\text{else } \vec{u}_i(g+1) := \vec{u}_i(g);\ i = 1,...,\mu$$

Several DE algorithms can be identified according to their type of mutation ($x$), number of difference vectors ($y$) and type of crossover ($z$). Commonly, the notation $DE/x/y/z$ is used to name a DE algorithm. Where $x$, means the way the vector to be mutated is chosen, $y$ indicates the number of difference vectors is used, and $z$ is the type of differential crossover implemented. For instance, the previously described algorithm is known as the $DE/rand/1/bin$, which means than the to be mutated vector is selected randomly, only one difference vector is calculated and the scheme of crossover is binomial. In general $x \in \{rand, best, current-to-rand\}$, $y \in \{1,2,...,n\}$, and $z \in \{bin, \exp\}$.

**Extensions of DE and a smoother operator**

Since originally DE algorithms were designed to solve unconstrained static optimization problems, a modification is required in order to deal with constraints for the controls. A clipping technique has been introduced to guarantee that only feasible trial vectors are generated after the mutation and crossover operators:

$$u'_{ji}(g) = \begin{cases} \beta_j \ \ if \ \ u'_{ji}(g) > \beta_j \\ \alpha_j \ \ if \ \ u'_{ji}(g) < \alpha_j \end{cases} ; \ j = 1,2,...,d; i = 1,2,...,\mu \tag{7}$$

where $\alpha_j$ and $\beta_j$ represent the lower and upper boundaries of the control variables, respectively. A smoother operator is defined according to [8] as follows:

$$u_{j,i} = median(u_{j-F,i}, u_{j-F+1,i}, ..., u_{j,i}, ..., u_{j+F-1,i}, u_{j+F,i}) \tag{8}$$
$$j = F+1, F+2, ..., N-F \ ; i = 1,2,...,\mu$$

where $F = 1,2,..$ is the filtering radius. Both Differential Evolution algorithms and its extensions were programmed as an m-file in the Matlab environment.

## 4 Multimodal Optimal Control of Bifunctional Catalyst Blend

A chemical process converting methylcyclopentane to benzene in a tubular reactor is modeled by a set of seven differential equations:

$$\dot{x}_1 = -k_1 x_1 \ . \tag{9}$$

$$\dot{x}_2 = k_1 x_1 - (k_2 + k_3) x_2 + k_4 x_5 \ . \tag{10}$$

$$\dot{x}_3 = k_2 x_2 \ . \tag{11}$$

$$\dot{x}_4 = -k_6 x_4 + k_5 x_5 \ . \tag{12}$$

$$\dot{x}_5 = k_3 x_2 + k_6 x_4 - (k_4 + k_5 + k_8 + k_9) x_5 + k_7 x_6 + k_{10} x_7 \ . \tag{13}$$

$$\dot{x}_6 = k_8 x_5 - k_7 x_6 \ . \tag{14}$$

$$\dot{x}_7 = k_9 x_5 - k_{10} x_7 \ . \tag{15}$$

where $x_i, i = 1,...,7$ are the mole fractions of the chemical species, and the rate constants ( $k_i$ ) are cubic functions of the catalyst blend $u(t)$:

$$k_i = c_{i1} + c_{i2} u + c_{i3} u^2 + c_{i4} u^3 \ ; \ i = 1,...,10 \tag{16}$$

The values of the coefficients $c_{ij}$ are given in [7]. The upper and lower bounds on the mass fraction of the hydrogenation catalyst are: $0.6 \le u(t) \le 0.9$ , and the initial vector

of mole fraction is $\mathbf{x}[0] = \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^T$. This is a continuous process operated in steady state, so that 'time' in equations (9)-(16) is equivalent to travel time and thus length along the reactor. The optimal control problem is to find the catalyst blend along the length of the reactor, which in the control problem formulation is considered at times $0 \le t \le t_f$ where the final effective residence time $t_f = 2000 g \cdot h / mol$ such that the concentration in the reactor is maximized: $J = x_7(t_f) \times 10^3$. Esposito and Floudas [16] found recently 300 local minima of this problem, so this is a challenging multimodal optimal control problem.

## 5 Singular Optimal Control of the Park-Ramirez Bioreactor

One optimal control problem that has a singular optimal solution was used to test the modified DE algorithm [8]. In this problem the goal is to maximize the production of protein. The system is described by the following differential equations:

$$\dot{x}_1 = g_1(x_2 - x_1) - \frac{x_1}{x_5}u \ . \quad (17)$$

$$\dot{x}_2 = g_2 x_3 - \frac{x_2}{x_5}u \quad (18)$$

$$\dot{x}_3 = g_3 x_3 - \frac{x_3}{x_5}u \ . \quad (19)$$

$$\dot{x}_4 = -g_4 g_3 x_3 - \frac{m - x_4}{x_5}u \ . \quad (20)$$

$$\dot{x}_5 = u \ . \quad (21)$$

where $g_1 = \frac{4.75 g_3}{0.12 + g_3}$, $g_2 = \frac{21.88 x_4}{(x_4 + 0.4)(x_4 + 62.5)}$, $g_3 = \frac{x_4 \exp(-5.01 x_4)}{0.10 + x_4}$, $g_4 = 58.75 g_2^2 + 1.71$.

The state variable $x_1$ represents amount of secreted protein [unit culture volume $L^{-1}$], $x_2$ denotes the total protein amount [unit culture volume $L^{-1}$], $x_3$ means culture cell density [g $L^{-1}$], $x_4$ culture glucose concentration [g $L^{-1}$], and $x_5$ the culture volume [L]. The control $u(t)$ represents the rate at which glucose is fed into the reactor [$Lh^{-1}$]. The secretion rate constant is given by $g_1$, the protein expression rate is calculated by $g_2$, the specific growth rate by $g_3$ and the biomass to glucose yield is estimated by $g_4$. The optimal control problem consists in the maximization of the amount of the secreted protein in a given time $t_f = 15h$. Therefore the performance index is given by $J = x_1(t_f)x_5(t_f)$. The control input satisfying the constraints $0 \le u(t) \le 2.5$ and the system initial conditions are $x(0) = [0,0,1.0,5.0,1.0]$. The dynamic model (eqns. 10-14) was programmed in the Matlab-Simulink environment. A C-MEX file

containing the dynamic equations was implemented in order to speed up the simulations. A variable step size Runge-Kutta integration method with a relative tolerance of $1e-8$ was applied. The DE algorithm was initialized randomly from the control's domain. Since DE algorithms are probabilistic methods the optimizations were repeated 10 times. The problem was solved for two number of variables N=50 and N=100.

## 6 Results and Discussion

**Multimodal optimal control problem**

Ten differential evolution algorithms were evaluated in solving the multimodal optimal control problem aforementioned. **Table 1** shows main results. NP is the population size used in each algorithm.

**Table 1.** Evaluation of several DE algorithms in solving a multimodal continuous-time optimal control problem

| DE | CR | F | NP | Kp | F.E. | STD | J* | STD |
|---|---|---|---|---|---|---|---|---|
| *DE* / *best* / 2 / *bin* | 0.0 | 0.9 | 15 | - | 2529 | 262.98 | 10.0942 | 0.0 |
| *DE* / *best* / 2 / exp | 0.0 | 0.9 | 20 | - | 3426 | 388.85 | 10.0942 | 0.0 |
| *DE* / *rand* / 1 / exp | 0.0 | 0.9 | 15 | - | 2289 | 295.31 | 10.0942 | 0.0 |
| *DE* / *rand* / 1 / *bin* | 0.0 | 0.9 | 20 | - | 3044 | 351.22 | 10.0942 | 4e-5 |
| *DE* / *rand* / 2 / *bin* | 0.0 | 0.9 | 20 | - | 3872 | 332.69 | 10.0942 | 4e-5 |
| *DE* / *rand* / 2 / exp | 0.0 | 0.9 | 20 | - | 3882 | 440.44 | 10.0942 | 0.0 |
| *DE* / *curr – to – rand* / 1 / *bin* | 0.0 | 0.9 | 15 | 1 | 2100 | 346.04 | 10.0942 | 5e-5 |
| *DE* / *curr – to – rand* / 1 / exp | 0.0 | 0.9 | 15 | 1 | 2257 | 202.63 | 10.0942 | 0.0 |
| *DE* / *best* / 1 / *bin* | 0.0 | 0.9 | 25 | - | 3112 | 324.30 | 10.0942 | 0.0 |
| *DE* / *best* / 1 / exp | 0.0 | 1.0 | 25 | - | 3245 | 413.11 | 10.0941 | 2e-4 |

Results of table 1 represent the average of 10 runs regarding number of function evaluations (FE) and the objective function ($J^*$). A measure of population convergence was defined as a difference between worst and best solution satisfied a given value. In this case the accuracy required was 1e-3. Clearly all DE algorithms found

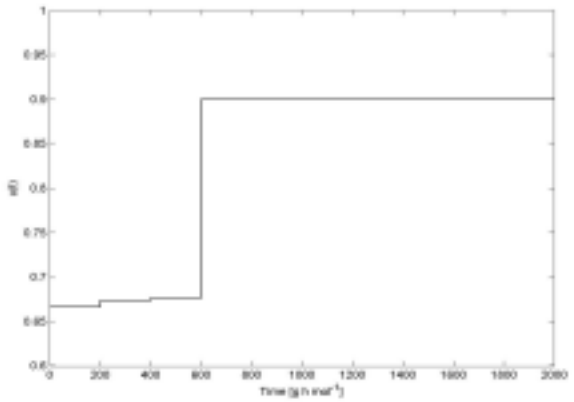


**Figure 1.** Optimal control trajectory of multimodal probem.

the global optimum with the given values of the parameters. Notice that because of the high multimodality of the problem the mutation parameter is greater and in some cases the population size was increased more than two times the size of the number of variables to be optimized. Price [11], suggests populations sizes between $2n$ and $20n$ but our results show that even with lower sizes DE algorithms can solve multimodal optimal control problems. Figure 1 shows the optimal control trajectory found by DE algorithms.

## Singular optimal control of Park-Ramirez bioreactor

Since DE algorithms are very robust it is easy to determine a set of parameters that provides an acceptable solution. Furthermore, the solved optimal control problem has likely only one solution so it was found that an almost standard setting worked out properly. In contrast to the commonly applied approach, which is based on the use of a too large population size, in our situation population size was chosen equal to the dimension of the optimization (N) problem. Since we did not expect a multimodal problem then the mutation constant was kept reasonably small. However, the crossover parameter was substantially increased in order to speed up the convergence of the algorithms. Table 2 and table 3 show the parameters settings (crossover constant, mutation parameter and population size) of $DE/rand/1/bin$ applied on optimal control problem using the Park-Ramirez bioreactor. Also the main results of the comparison regarding number of generations required, the number of function evaluations needed and the cost function value are presented.

**Table 2.** Results obtained by DE and smoother DE in solving a singular optimal control problem (Number of variables N=50)

| | CR | F | μ | Generations | Function Evaluations | $\mathbf{J}^*$ |
|---|---|---|---|---|---|---|
| **DE** | 0.9 | 0.6 | 50 | 5192 | 259600 | 32.41 |
| **SDE** | 0.9 | 0.6 | 50 | 932 | 46600 | 32.41 |

**Table 3.** Results obtained by DE and smoother DE in solving a singular optimal control problem (Number of variables N=100).

| | CR | F | μ | Generations | Function Evaluations | $\mathbf{J}^*$ |
|---|---|---|---|---|---|---|
| **DE** | 0.9 | 0.6 | 100 | 8251 | 825100 | 32.47 |
| **SDE** | 0.9 | 0.6 | 100 | 436 | 43600 | 32.47 |

Figure 2 and figure 3 show the optimal control trajectories calculated by both the DE and the smoother DE algorithms for N=50 and N=100 number of variables. In both cases the trajectories resulted on the same cost function value. Clearly, the trajectory generated by DE algorithm with a smoother operator has less oscillation than that obtained by DE. The oscillation of optimal control trajectory obtained by DE was as the control was parameterized more variables (N=100). A comparison of figures 2 and 3 makes apparent that only small differences can be distinguished between the optimal control trajectories calculated by the smother DE algorithm. The performance index values obtained for both situations N=50 and N=100 were exactly the same reported by [8] using Iterative Dynamic Programming. The improvement in efficiency according to the number of function evaluations as N=50 is used was 6 % in case of N=100

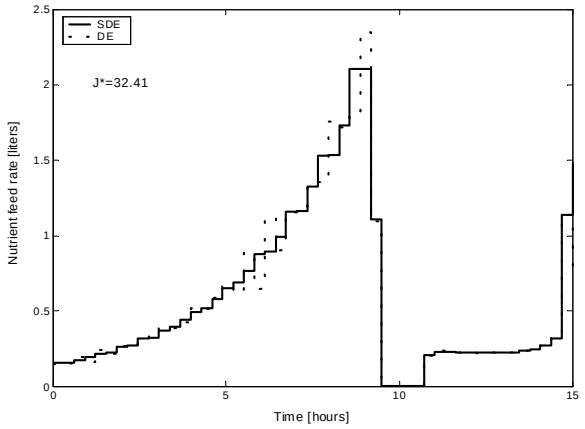



**Figure 2.** Optimal control trajectory of DE without and with smoother operator (N=50).

this was 19%. Similar percentages were obtained taking into consideration the number of generations. The explanation of this fact could be the increment of population size to $\mu = 100$ individuals as N=100. But also it is clear that avoiding the oscillation of the optimal trajectories speed up the convergence of the DE algorithm. Therefore, it is clear that using the smoother operator together with other DE operators, the performance of DE algorithms is improved considerably and also higher oscillation of optimal control trajectories can be avoided.

## 7 Conclusions

A highly multimodal optimal control problem was used to test the performance of several differential evolution algorithms. Results show that DE algorithms are good candidates to solve this class of problems since even using small populations they can find the global optimum trajectory. DE algorithms are robust and their parameters are chosen in a straightforward way. A smoother operator was proposed and evaluated in

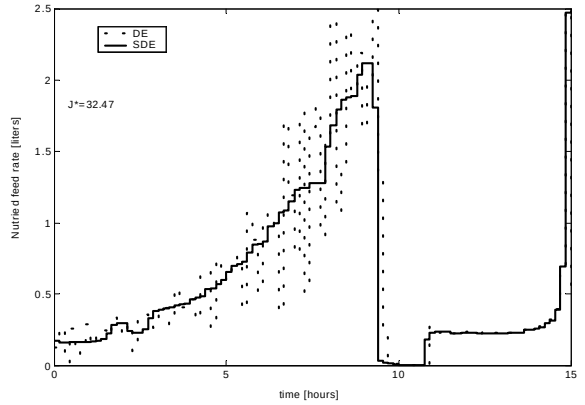



**Figure 3**. Optimal control trajectory without and with smoother operator N=100.

solving singular optimal control problems by Differential Evolution algorithms. The evaluation of the smoother operator on a dynamic optimization problem of a nonlinear bioreactor showed that the operator not only removed the oscillation of the optimal control trajectory, but also it speed up the convergence of the of a DE algorithm.

## References

1. Park, S., Ramirez, W. F.: Optimal production of secreted protein in fed-batch reactors. AIChE Journal 34 (9) (1988) 1550-1558.
2. Park, S., Ramirez, W. F.: Dynamics of foreign protein secretion from *Saccharomyces cerevisiae*. Biotechnology and Bioengineering 33 (1989) 272-281.
3. Menawat, A., Mutharasan, R., Coughanowr, D. R.: Singular optimal control strategy for a fed-batch bioreactor: numerical approach. AIChE Journal 33 (5) (1987) 776-783.
4. Roubus, J.A., de Gooijer, C.D., van Straten, G., van Boxtel, A.J.B.: Comparison of optimization methods for fed-batch cultures of hybridoma cells, Bioproc. Eng.17 (1997) 99-102.
5. Roubos, J.A., van Straten, G., van Boxtel, A.J.B.. An evolutionary strategy for fed-batch bioreactor optimization:concepts and performance, Journal of Biotechnology 67 (1999) 173-187.
6. Luus, R.. On the application of Iterative Dynamic Programming to Singular Optimal Control problems. IEEE, Transactions on Automatic Control 37 (11) (1992)
7. Luus, R. Iterative Dynamic Programming. Chapman & Hall/CRC., Boca Raton, (2000).
8. Tholudur, A., Ramirez, W.F.: Obtaining smoother singular arc policies using a modified iterative dynamic programming algorithm. International Journal of Control, 68(5) (1997)1115-1128.
9. Salomon, R.: Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. BioSystems 39 (1996) 263-278.
10. Storn, R., Price, K.: Differential Evolution- a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11 (1997) 341-359.
11. Price K., V.: An Introduction to Differential Evolution. In Corne, D., Dorigo, M., and Glover, F. (eds.). New Ideas in Optimization. Mc Graw Hill (1999).
12. Lopez-Cruz, I.L. van Willigenburg, G., van Straten G. 2003. Efficient evolutionary algorithms for multimodal optimal control problems. Journal of applied soft computing 3(2): 97-122.
13. Moles, C.G., Banga, J.R., Keller, K. 2004. Solving nonconvex climate control problems: pitfalls and algorithm performances. Journal of applied soft computing 5:35-44.
14. Kirk, D.E.: Optimal control theory. An introduction. Dover Publications, Inc. New York. (1998).
15. Bryson, A.E.: Dynamic Optimization. Addison-Wesley. (1999).
16. Esposito W.R., Floudas, Ch.A.: Deterministic global optimisation in nonlinear optimal control problems. Journal of global optimisation 17 (2000) 97-126.

# Author Index

Índice de autores


Arai, Masahiko 23
Ayala Ramirez, Victor 111
Babaian, Tamara 65
Barrón, Ricardo 21,131,141
Belaid, Benhamou 45
Boughaci, Dalila 5
Brena, Ramon 9
Carlin, Alan 5
Castiello, Ciro 53
Chandra, Priti 5
Cruz, Benjamín 21
De la Calleja Manzanedo, René 35
Díaz de León, Juan Luis 261
Fakih, Adel 163
Fanelli, Anna Maria 153
Feng, Boqin 195
Ferrández, Oscar 219
González Mendoza, Miguel 175
Guillen, Rocio 229
Habiba, Drias 45
Hannon, Charles 239
Hernández Gress, Neil 175
Hui, Zhang 13
Kim, Inkyeom 185
Kozareva, Zornitsa 219
Kuri Morales, Angel 207
Kuroiwa, Shingo 3
Leduc, Luis A. 251
Li, Bo 195
Liu, Qiong 3
Longoria Mendez, Cruz A. 111
Lopez Cruz, Irineo L. 271
Lu, Jun 195
Lu, Xin 3
Marin, Cesar 79
Martínez Ríos, Félix Orlando 89
Mejía Guevara, Iván 207
Mendez, Gerardo M. 251
Montoyo, Andrés 219
Mora Vargas, Jaime 175
Muñoz, Rafael 219
Pujari, Arun K. 55
Ren, Fuji 3
Rodríguez Lucatero, Carlos 89
Rojano Aguilar, Abraham 271
Saade, Jean 163
Sanchez Yanez, Raul E. 111
Schmolze, James G. 65
SiKun, Li 13
Sossa, Humberto 121,131,141
Terashima Marín, Hugo 35
Torres, Benjamín 141
Umre, Ashish 99
Valenzuela Rendón, Manuel 35
Vázquez, Fernando 261
Vázquez, Roberto A. 131
Wakeman, Ian 99
Yun, Minyoung 185

# Editorial Board of the Volume

Comité editorial del volumen

Gabriela Ochoa Meier
Mauricio Osorio Galindo
Andrés Pérez Uribe
Manuel Palomar
Luis Alberto Pineda
Andre Ponce de Leon F. de Carvalho
David Poole
Bhanu Prasad
Jorge Adolfo Ramírez Uresti
Fernando Ramos
Carlos Alberto Reyes García
Abdennour El Rhalibi
Maria Cristina Riff
Roger Z. Rios
Dave Robertson
Horacio Rodríguez
Riccardo Rosati
Isaac Rudomín
Alessandro Saffiotti
Gildardo Sánchez
Alberto Sanfeliú Cortés
Andrea Schaerf
Thomas Schiex
Leonid Sheremetov
Grigori Sidorov
Carles Sierra
Alexander V. Smirnov
Maarten van Someren
Juan Humberto Sossa Azuela
Rogelio Soto
Thomas Stuetzle
Luis Enrique Sucar Succar
Ricardo Swain Oropeza
Hugo Terashima
Demetri Terzopoulos
Manuel Valenzuela
Juan Vargas
Felisa Verdejo
Manuel Vilares Ferro
Toby Walsh
Alfredo Weitzenfeld
Nirmalie Wiratunga
Franz Wotawa
Kaori Yoshida
Claus Zinn
Berend Jan van der Zwaag

# Additional Reviewers

Árbitros adicionales

Juan C. Acosta Guadarrama
Héctor Gabriel Acosta Mesa
Teddy Alfaro
Miguel A. Alonso
José Ramón Arrazola
Stella Asiimwe
Séverine Bérard
Fco. Mario Barcala Rodríguez
Axel Arturo Barcelo Aspeitia
Adam D. Barker
Alejandra Barrera
Gustavo E. A. P. A. Batista
Abderrahim Benslimane
Arturo Berrones
Bastian Blankenburg
Pascal Brisset
Andreas Bruening
Mark Buckley
Olivier Buffet
Diego Calvanese
Hiram Calvo
Niccolo Capanni
Carlos Castillo
Sutanu Chakraborti
Carlos Chesñevar
Federico Chesani
Wu Feng Chung
Murilo Coelho Naldi
Mark Collins
Jean François Condotta
Miguel Contreras
Sylvie Coste Marquis
Anne Cregan
Ronaldo Cristiano Prati
Juan A. Díaz
Víctor Manuel Darriba Bilbao
Michael Dekhtyar
Deepak Devicharan
Luca Di Gaspero
Marissa Diaz
Luigi Dragone
Edgar Duéñez
Mehmet Önder Efe
Arturo Espinosa Romero
Katti Faceli
Antonio Fernandez Caballero
Antonio Ferrández
Armin Fiedler
Alfredo Gabaldón
Arturo Galván Rodríguez
Ariel García
Cormac Gebruers
Karina Gibert
Andrea Giovannucci
Fernando Godínez
Giorgi Goguadze
Miguel González
Jorge Graña
Federico Guedea
Alejandro Guerra Hernández
Daniel Gunn
Everardo Gutiérrez
Christian Hahn
Emmanuel Hebrard
Benjamín Hernández
Martin Homik
Rodolfo Ibarra
Boyko Iliev
Bartosz Jablonski
Jean Yves Jaffray
Sylvain Jasson
Daniel Jolly
Narendra Jussien
Lars Karsson
Ryszard Klempous
Jerzy Kotowski
A. Krizhanovsky
Juan Carlos López Pimentel
David Lambert
Darío Landa Silva
Jérme Lang
Huei Diana Lee
Domenico Lembo
Paul Libberecht
Ana Carolina Lorena
Robert Lothian

Henryk Maciejewski
Fernando Magan Muñoz
Michael Maher
Donato Malerba
Salvador Mandujano
Ana Isabel Martinez Garcia
Patricio Martinez Barco
Jarred McGinnis
Andreas Meier
Manuel Mejia Lavalle
Corrado Mencar
Thomas Meyer
Thomas Meyer
Erik Millan
Monica Monachini
Rebecca Montanari
Andrés Montoyo
Jaime Mora Várgas
José Andrés Moreno Pérez
Rafael Muñoz
Martin Muehlenbrock
Rahman Mukras
Amedeo Napoli
Gonzalo Navarro
Adeline Nazarenko
Juan Carlos Nieves
Peter Novak
Slawomir Nowaczyk
Oscar Olmedo Aguirre
Magdalena Ortiz de la Fuente
María Osorio
Joaquín Pacheco
Marco Patella
Jesús Peral
Mats Petter Pettersson
Steven Prestwich
Bernard Prum
José Miguel Puerta Callejón
Alonso Ramírez Manzanárez
Fernando Ramos
Orión Fausto Reyes Galaviz
Francisco Ribadas Peña
Fabrizio Riguzzi
Leandro Rodríguez Liñares
Juan A. Rodríguez Aguilar
Raquel Ros
Maximiliano Saiz Noeda
S. Sandeep
P. Sanongoon
Cipriano Santos
Vitaly Schetinin
Marvin Schiller
Przemyslaw Sliwinski
Jasper Snoek
Thamar Solorio
Claudia Soria
Eduardo J. Spinosa
Cyrill Stachniss
Ewa Szlachcic
Armagan Tarim
Choh Man Teng
Paolo Torroni
Elio Tuci
Carsten Ullrich
L. Alfonso Ureña López
Diego Uribe
Mars Valiev
Maria Vargas Vera
Wamberto Vasconcelos
José Luis Vega
José Luis Vicedo
Jesús Vilares Ferro
Mario Villalobos Arias
Nic Wilson
Sean Wilson
Claudia Zepeda
Juergen Zimmer

Impreso en los Talleres Gráficos
de la Dirección de Publicaciones
del Instituto Politécnico Nacional
Tresguerras 27, Centro Histórico, México, D.F.
Noviembre de 2005.
Printing 500 / Edición 500 ejemplares.