Lecture Notes in Artificial Intelligence 1793

Subseries of Lecture Notes in Computer Science
Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science
Edited by G.Goos, J. Hartmanis, and J. van Leeuwen

Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Osvaldo Cairo L. Enrique Sucar
Francisco J. Cantu (Eds.)

MICAI 2000: Advances in Artificial Intelligence

Mexican International Conference
on Artificial Intelligence
Acapulco, Mexico, April 11-14, 2000
Proceedings

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Osvaldo Cairo
Instituto Tecnologico Autonomo de Mexico (ITAM)
Department of Computer Science
Rio Hondo 1, Progreso Tizapan, 01000 Mexico D.F.
E-mail: cairo@lamport.rhon.itam.mx

L. Enrique Sucar
Monterrey Institute of Technology (ITESM)
Campus Morelos, Av. Reforma 182-A, Lomas de Cuernvaca, Temixco
Morelos, 62589, Mexico
E-mail: esucar@campus.mor.itesm.mx

Francisco J. Cantu
Monterrey Institute of Technology (ITESM)
Center for Artificial Intelligence
Avenida Eugenio Garza Sada 2501, Monterrey N.L., 64849 Mexico
E-mail: fcantu@campus.mty.itesm.mx

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Advances in artificial intelligence : proceedings / MICAI 2000,
Mexican International Conference on Artificial Intelligence, Acapulco,
Mexico, April 11 - 14, 2000. Osvaldo Cairo ... (ed.). - Berlin ;
Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ;
Paris ; Singapore ; Tokyo : Springer, 2000
(Lecture notes in computer science ; Vol. 1793 : Lecture notes in
artificial intelligence)
ISBN 3-540-67354-7

CR Subject Classification (1991): I.2

ISSN 0302-9743
ISBN 3-540-67354-7 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by PTP Berlin, Stefan Sossna
Printed on acid-free paper SPIN: 10720076 06/3142 5 4 3 2 1 0

Preface

Fifty years ago, A. Turing predicted that by 2000 we would have a machine that could pass the Turing test. Although this may not yet be true, AI has advanced significantly in these 50 years, and at the dawn of the XXI century is still an active and challenging field. This year is also significant for AI in Mexico, with the merging of the two major AI conferences into the biennial Mexican International Conference on Artificial Intelligence (MICAI) series.

MICAI is the union of the Mexican National AI Conference (RNIA) and the International AI Symposium (ISAI), organized annually by the Mexican Society for AI (SMIA, since 1984) and by the Monterrey Institute of Technology (ITESM, since 1988), respectively. The first Mexican International Conference on Artificial Intelligence, MICAI 2000, took place April 11-14, 2000, in the city of Acapulco, Mexico. This conference seeks to promote research in AI, and cooperation among Mexican researchers and their peers worldwide. We welcome you all.

Over 163 papers from 17 different countries were submitted for consideration to MICAI 2000. After reviewing them thoroughly, MICAI's program committee, referees, and program chair accepted 60 papers for the international track. This volume contains the written version of the papers and invited talks presented at MICAI.

We would like to acknowledge the support of the American Association for Artificial Intelligence (AAAI), and the International Joint Conference on Artificial Intelligence (IJCAI). We are specially grateful for the warm hospitality and generosity offered by the Acapulco Institute of Technology.

Support for MICAI was provided in part by CONACYT's Directorate of the Research and Development on Information Science Network, the Mexican Society for Computer Science, and the following universities and research institutions: BUAP, CICESE, CINVESTAV, IPN-CIC, ITAM, ITESM, LANIA, UDLA, UNAM-IIMAS, and UV.

We gladly acknowledge our invited speakers Wolfgang Wahlster, Hector Levesque, Jay Liebowitz, Adolfo Guzman, Jose Luis Marroquin, and George L. Lasken. A special word of thanks goes to the members of the advisory board, the members of the program committee, the reviewers, our corporate sponsors, our support staff and to Patricia Mora from JTESM for editing this volume.

Last but not least, we warmly thank all the speakers for their excellent lectures.

Acapulco, Mexico, April 11, 2000

Osvaldo Cairo
L. Enrique Sucar
Francisco J. Cantu

Organization

MICAI 2000 was organized by the Mexican Society for Artificial Intelligence and the Acapulco Institute of Technology (ITA).

Conference Committee

Conference Chair:	Francisco J. Cantu (ITESM-Mty, Mexico)
	L. Enrique Sucar (ITESM-Mor, Mexico)
Program Chair:	Osvaldo Cairo (ITAM, Mexico)
	Luis A. Pineda (UNAM, Mexico)
Tutorials:	Marc Boumedine (ITESM-CCM, Mexico)
Workshops:	Humberto Sossa (IPN, Mexico)
Local Arrangements:	Blanca E. Lopez (ITA)
	Juan M. Rodriguez (ITA)
Publicity and Design:	Moraima Campbell (ITESM-Mty, Mexico)
Finance and Registration :	Leticia Rodriguez (ITESM-Mty, Mexico)

Advisory Committee

Felipe Bracho
Alan Bundy
Ofelia Cervantes
Anthony Cohn
Francisco Garijo
Randy Goebel
Adolfo Guzman
Robert de Hoog
Felipe Lara
Christian Lemaitre
Jay Liebowitz
Cristina Loyo
Donald Michie
Jose Negrete
Pablo Noriega
Judea Pearl
Antonio Sanchez
Xindong Wu
Wolfgang Wahlster
Carlos Zozaya

Program Committee

Leopoldo Altamirano
Matias Alvarado
Ronald Arkin
Gerardo Ayala
Richard Benjamin
Marc Boumedine
Carlos Coello
Helder Coelho
Rogelio Davila
Javier Diez
Robert de Hoog
Guillermo de Ita
Boris Escalante
Vladimir Estivill-Castro
Jesus Favela
Asuncion Gomez-Perez
Jose Luis Gordillo
Silvia Guardati
Adolfo Guzman Arenas
Leo Joskowicz
Natalia Juristo
Nicolas Kemper
Angel Kuri
Ana Maria Martinez

Manuel Martinez
Raul Monroy
Eduardo Morales
Guillermo Morales Luna
Santiago Negrete
Alberto Oliart Ros
Mauricio Osorio
Luis Pineda
Alexandro Provetti
Carlos Ramirez
Homero Rios
Jaime Siamo Sichman
Carlos Sierra
Rogelio Soto
L. Enrique Sucar
Manuel Valenzuela
Javier Vega
Marteen Von Someren
Toby Walsh
Alfredo Weitzenfeld Zdnek Zdrahal
Carlos Zozaya

Referees

Angelica Antonio
Josep-Lluis Arcos
Eva Armengol
Oscar Dieste
Pere Garcia
Mario Martin
Nelson Medinilla
Pedro Meseguer
Ana Maria Moreno
Andres Silva
Sira Vegas

Collocated Conferences

3^o TAINA - Workshop on AI
TIARP - Iberoamerican Workshop on Pattern Recognition

Sponsoring Institutions

The American Association for Artificial Intelligence,
International Joint Conference on Artificial Intelligence,
The Mexican Society for Computer Science,
CONACYT REDII.

And support from

ITESM, ITAM, IPN-CIC, UDLA, CICESE, UNAM-IIMAS, LANIA, UV, BUAP
Corporate support from BARTEC, DigiPro, World Software Services, Softek, Microsoft, IBM.

We would like to thank Francisco Solsona, Julio Barreiro, Jose Galvez, and Erick Rodriguez for their excellent organizational support. Special thanks go to Patricia Mora and the Springer staff for editing this volume.

Table of Contents

Knowledge Representation and Reasoning

Robotics and Computer Vision

Machine Learning

Neural Networks

Knowledge-Based Systems

Speech Recognition and Natural Language

Multiagent Systems

Reasoning Under Uncertainty

Intelligent Tutoring Systems

Searching for a Solution to Program Verification=Equation Solving in CCS*

Raúl Monroy[1], Alan Bundy[2], and Ian Green[2]

[1] Computer Science Department, ITESM Campus Estado de México
Apdo. Postal 50, Módulo de Servicio Postal Campus Edo. de México del ITESM,
52926 Atizapán, Edo. de México, México,
raulm@campus.cem.itesm.mx
[2] Division of Informatics, The University of Edinburgh,
80 South Bridge, EH1 1HN, Scotland, U.K,
{A.Bundy, I.Green}@ed.ac.uk

Abstract. Unique Fixpoint Induction, UFI, is a chief inference rule to prove the equivalence of recursive processes in CCS [7]. It plays a major role in the equational approach to verification. This approach is of special interest as it offers theoretical advantages in the analysis of systems that communicate values, have infinite state space or show parameterised behaviour.
The use of UFI, however, has been neglected, because automating theorem proving in this context is an extremely difficult task. The key problem with guiding the use of this rule is that we need to know fully the state space of the processes under consideration. Unfortunately, this is not always possible, because these processes may contain recursive symbols, parameters, and so on.
We introduce a method to automate the use of UFI. The method uses middle-out reasoning and, so, is able to apply the rule even without elaborating the details of the application. The method introduces variables to represent those bits of the processes' state space that, at application time, were not known, hence, changing from equation verification to equation solving.
Adding this method to the equation plan developed by Monroy, Bundy and Green [8], we have implemented an automated verification planner. This planner increases the number of verification problems that can be dealt with fully automatically, thus improving upon the current degree of automation in the field.

1 Introduction

The Calculus of Communicating Systems [7] (CCS) is an algebra suitable for modelling and analysing processes. CCS is well-established in both industry and academia and has strongly influenced the design of LOTOS [5].

* The authors are supported in part by grants CONACyT-REDII w/n and EPSRC GR/L/11724.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 1–12, 2000.

Unique Fixpoint Induction (UFI) is an inference rule for reasoning about recursive processes in CCS and other process algebras [7]. UFI plays a major role in the equational approach to verification. This approach is of special interest as it offers theoretical advantages in the analysis of systems that communicate values, have infinite state space or show parameterised behaviour.

The use of UFI has been neglected. This is because automating theorem proving in this context is an extremely difficult task. The key problem with guiding the use of this rule is that we need to know fully the state space of the agents under consideration. Unfortunately, this is not always possible, for these agents may contain recursive symbols, parameters, and so on. We suggest that a proof planning approach can provide significant automation in this context.

Proof planning [1] is a meta-level reasoning technique. A *proof plan* captures general knowledge about the commonality between the members of a proof family, and is used to guide the search for more proofs in that family. We introduce a proof plan to automate the use of UFI, based on middle-out reasoning.

Middle-out reasoning (MOR) is the use of meta-variables to represent unknown, 'eureka' values. These meta-variables are gradually refined as further proof steps take place. Control of this refinement is the key in the success of this approach and proof planning provides the necessary means to guide the selection of instantiations and proof steps.

Using MOR, we can apply UFI even without elaborating the details of the application. We approach a verification problem by generalising the input conjecture in a way that the use of UFI becomes immediate. If successful, the proof of the generalised goal is then used to justify the initial one. To generalise the input conjecture, we create a new process family and replace it for the process under verification. The new process family and the specification have similar size and structure, hence facilitating the use of UFI. The new process familiy is under specified, because — via MOR — it contains meta-variables representing those bits of the process's state space that, at introduction time, were not known. We therefore change from equation verification to equation solving.

The proof plan presented here is an extension of the equational verification plan developed by Monroy, Bundy and Green [8]. The proof plan contains three additional strategies, with which it fully automatically decides when and how to apply UFI, as well as driving the solution of equations. The proof plan increases the number of verification problems that can be dealt with fully automatically, thus improving upon the current degree of automation in the field.

Overview of Paper In §2 we describe CCS and UFI. In §3 we characterise the kinds of proofs we shall automate, highlighting search control issues. In §4 we describe proof planning and the verification plan. In §5, §6 and §7 we introduce a method for guiding the use of UFI, the generalisation of the input goal and equation solving. We illustrate aspects of this method with a running example in §8. Finally, we summarise experimental results, discuss related work, as well as drawing conclusions in §9.

2 CCS

Terms of CCS represent agents. Agents are circumscribed by their entire capabilities of interaction. Interactions occur either between two agents, or between an agent and its environment. These communicating activities are referred to as *actions*. An action is said to be *observable*, if it denotes an interaction between an agent and its environment, otherwise it is said to be *unobservable*. This interpretation of observation underlies a precise and amenable theory of behaviour: whatever is observable is regarded as the behaviour of a system. Two agents are held to be equivalent if their behaviour is indistinguishable to an external observer.

2.1 Syntax and Semantics

The set of Actions, $\mathcal{A}ct = \{\alpha, \beta, \ldots\}$, contains the set of *names*, $\mathcal{A}$, the set of *co-names*, $\overline{\mathcal{A}}$, and the unobservable action τ, which denotes process intercommunication. $\mathcal{A}$ and $\overline{\mathcal{A}}$ are both assumed to be countable and infinite. Let $a, b, c, \ldots$ range over $\mathcal{A}$, and $\overline{a}, \overline{b}, \overline{c}, \ldots$ over $\overline{\mathcal{A}}$. The set of labels, $\mathcal{L}$, is defined to be $\mathcal{A} \cup \overline{\mathcal{A}}$; hence, $\mathcal{A}ct = \mathcal{L} \cup \{\tau\}$. Let $\ell, \ell', \ldots$ range over $\mathcal{L}$. Let K, $\overline{K}$, L, $\overline{L}$, ... denote subsets of $\mathcal{L}$.

$\mathcal{K}$ is the set of agent constants, which refer to unique behaviour and are assumed to be declared by means of the definition facility, $\stackrel{\text{def}}{=}$. Let $A, B, C, \ldots$ range over $\mathcal{K}$. Constants may take parameters. Each *parameterised constant* A with arity n is assumed to be given by a set of defining equations, each of which is of the form: $b \rightarrow A_{\langle x_1,\ldots,x_n\rangle} \stackrel{\text{def}}{=} E$. E does not contain free parameter variables other than $x_1, \ldots, x_n$. $\rightarrow$ denotes logical implication.

We assume parameter expressions, e, built from parameter variables $x, y, \ldots$, parameter constants $v_1, v_2, \ldots$ and any operators we may require, $\times, \div, \text{even}, \ldots$. We also assume boolean expressions, b, with similar properties except that they are closed under the logical connectives. Parameter constants might be of any type.

The set of *agent expressions*, $\mathcal{E}$, is defined by the following abstract syntax:

$$E ::= \; A \mid \alpha.E \mid \sum_{i \in I} E_i \mid E \mid E \mid \; E \setminus L \mid E[f]$$

where f stands for a relabelling function. Informally, the meaning of the *combinators* is as follows: *Prefix*, (), is used to convey discrete actions. *Summation*, ($\sum$), disjoins the capabilities of the agents E_i, ($i \in I$); as soon as one performs any action, the others are dismissed. Summation takes an arbitrary, possibly infinite, number of process summands. Here, Summation takes one of the following forms: i) The *deadlock agent*, $\mathbf{0}$, capable of no actions whatever; $\mathbf{0}$ is defined to be $\sum_{i \in \emptyset} E_i$; ii) *Binary Summation*, which takes two summands only, $E_1 + E_2$ is $\sum_{i \in \{1,2\}} E_i$; iii) *Indexed Summation over sets*, $\sum_{i \in \{e\} \cup I} E_i = E(e) + \sum_{i \in I} E_i$; and iv) *Infinite Summation over natural numbers*.

Parallel Composition, ($\big|$), is used to express concurrency: $E \mid F$ denotes an agent in which E and F may proceed independently, but may also interact with each other. The *Relabelling* combinator, ($[\,]$), is used to rename port labels: $E[f]$ behaves as E, except that its actions are renamed as indicated by f. *Restriction*, ($\backslash$), is used for internalising ports: $E \setminus L$ behaves like E, except that it cannot execute any action $\ell \in L \cup \overline{L}$.

Processes are given a meaning via the labelled transition system $(\mathcal{E}, \mathcal{A}ct, \stackrel{\alpha}{\rightarrow})$, where $\stackrel{\alpha}{\rightarrow}$ is the smallest transition relation closed under the following transition rules (the symmetric rule for $\mid$ has been omitted:)

$$\frac{}{or\alpha.E \stackrel{\alpha}{\rightarrow} E} \qquad \frac{E_j \stackrel{\alpha}{\rightarrow} E'}{or\sum_{i\in I} E_i \stackrel{\alpha}{\rightarrow} E'}\ (j \in I) \qquad \frac{E \stackrel{\alpha}{\rightarrow} E'}{orA \stackrel{\alpha}{\rightarrow} E'}\ (A \stackrel{\text{def}}{=} E) \qquad \frac{E \stackrel{\alpha}{\rightarrow} E'}{orE[f] \stackrel{f(\alpha)}{\rightarrow} E'[f]}$$

$$\frac{E \stackrel{\alpha}{\rightarrow} E'}{orE \mid F \stackrel{\alpha}{\rightarrow} E' \mid F} \qquad \frac{E \stackrel{\ell}{\rightarrow} E' \quad F \stackrel{\overline{\ell}}{\rightarrow} F'}{or\ E \mid F \stackrel{\tau}{\rightarrow} E' \mid F'} \qquad \frac{E \stackrel{\alpha}{\rightarrow} E'}{orE \setminus L \stackrel{\alpha}{\rightarrow} E' \setminus L}\ (\alpha, \overline{\alpha} \not\in L)$$

The interpretation of these rules should be straightforward and is not discussed here further.

Process (states) are related to each other: E' is a *derivative* of E, whenever $E \stackrel{\alpha}{\rightarrow} E'$. Similarly, E' is a *descendant* of E, whenever $E \stackrel{\alpha}{\Rightarrow} E'$, where $\stackrel{\alpha}{\Rightarrow}$ is given as $(\stackrel{\tau}{\rightarrow})^* \stackrel{\alpha}{\rightarrow} (\stackrel{\tau}{\rightarrow})^*$.

2.2 Unique Fixpoint Induction

Unique Fixpoint Induction (UFI) is a rule for reasoning about recursive processes [7]. UFI states that two processes are equivalent, if they satisfy the same set of (recursive) equations, so long as the set of equations has one, and only one, solution.

Uniqueness of solution of equations is guaranteed by two syntactic properties: guardedness and sequentiality. X is *guarded* in E if each occurrence of X is within some subexpression $\ell.F$ of E, for $\ell \in \mathcal{L}$. X is *sequential* in E if it occurs in E only within the scope of Prefix or Summation.

The notions of guardedness and sequentially are extended to sets of equations in the obvious way: the set of equations $\widetilde{X} = \widetilde{E}$ is guarded (respectively sequential) if $E_i (i \in I)$ contains *at most* the variables $X_i (i \in I)$ free, and these variables are all guarded (respectively sequential) in E_i.

Let the expressions $E_i\,(i \in I)$ contain at most the variables $X_i\,(i \in I)$ free, and let these variables be all guarded and sequential in each E_i. Then, the UFI inference rule is as follows:

$$\text{If } \widetilde{P} = \widetilde{E}\{\widetilde{P}/\widetilde{X}\} \text{ and } \widetilde{Q} = \widetilde{E}\{\widetilde{Q}/\widetilde{X}\}, \text{ then } \widetilde{P} = \widetilde{Q}$$

There is one aspect to the UFI rule that is worth mentioning: it reasons about process families. We cannot prove that some individuals satisfy a property without proving that such a property is satisfied by all. This is because the equations of each individual are incomplete and, by definition, to apply UFI the system of equations, $X_i = E_i (i \in I)$, must be such that each E_i contains *at most* the variables $X_i (i \in I)$ free.

3 Program Verification

We use CCS both as a programming language and as a specification language. Specifications are assumed to be explicitly given by means of C-declarations.

Definition 1 (C-declaration). *A set of definitions* $\widetilde{C} \to \widetilde{S} \stackrel{def}{=} \widetilde{E}\{\widetilde{S}/\widetilde{X}\}$ *is called a* C-declaration *if it satisfies the following conditions:*

1. *the expressions* $E_i\,(i \in I)$ *contain at most the variables* $X_i\,(i \in I)$ *free, and these variables are all guarded and sequential in each* E_i*;*
2. $S_i = S_j$ *implies* $i = j$*; and*
3. *the conditions* $C_i\,(i \in I)$ *are all fundamental, in that they cannot be given as the disjunction of two or more relations. For example,* $\geq$ *is not fundamental since* $x \geq y$ *means that* $x > y$ *or* $x = y$*.*

For example, to specify the behaviour of a buffer of size n, we write:

$$\begin{aligned} n \neq 0 \to \quad & Buf_{\langle n,0\rangle} \stackrel{\text{def}}{=} in.Buf_{\langle n,s(0)\rangle} \\ n > s(k) \to & Buf_{\langle n,s(k)\rangle} \stackrel{\text{def}}{=} in.Buf_{\langle n,s(s(k))\rangle} + \overline{out}.Buf_{\langle n,k\rangle} \\ n = s(k) \to & Buf_{\langle n,s(k)\rangle} \stackrel{\text{def}}{=} \overline{out}.Buf_{\langle n,k\rangle} \end{aligned}$$

where s stands for the successor function on natural numbers.

The systems under verification are arbitrary CCS terms. They reflect at the required level of detail all concurrency issues. What is more, they may contain recursive functions, which are used to capture the structure of one or more process subcomponents. We call these kinds of expressions *concurrent forms*. For example, we can implement a buffer of size n by linking n buffers of size 1:

$$\begin{array}{lll} n = 0 \;\to\; C^{s(n)} = C & \text{where} & C \stackrel{\text{def}}{=} in.D \quad D \stackrel{\text{def}}{=} \overline{out}.C \\ n \neq 0 \;\to\; C^{s(n)} = C^\frown C^{(n)} & & P^\frown Q \stackrel{\text{def}}{=} (P[c/out] \;\big|\; Q[c/in]) \setminus \{c\} \end{array}$$

Let P be a concurrent form and let S be a C-declaration. Then we call $P = S$ an instance of the *verification problem*. This is an example verification problem:

$$\forall n\!:\!\text{nat}.\; n \neq 0 \to C^{(n)} = Buf_{\langle n,0\rangle} \tag{1}$$

it states that a chain of n copies of a buffer of size one behaves as a buffer of size n.

We conclude this section noting that, since specifications are given, we need not use the general form of UFI. Instead we use the following, stronger rule:

$$\frac{\widetilde{P} = \widetilde{E}\{\widetilde{P}/\widetilde{X}\}}{\widetilde{P} = \widetilde{S}} \quad \widetilde{S} \stackrel{\text{def}}{=} \widetilde{E}\{\widetilde{S}/\widetilde{X}\} \tag{2}$$

We call $\widetilde{P} = \widetilde{E}\{\widetilde{P}/\widetilde{X}\}$ the *output equation set* and $\{\widetilde{P}/\widetilde{X}\}$ the *process substitution*.

Having given the verification problem, we now attempt to give the reader a flavour as to the difficulties of automating the search for a verification.

3.1 Search Control Problems within Program Verification

How should one proceed to verify (1)? Clearly, $Buf_{\langle n,k\rangle}$ and $C^{(n)}$ are both recursive and, so, they suggest the use of induction. However, $C^{(n)}$ suggests the use of a sort of induction other than UFI, namely: structural induction, or induction for short.

Induction and UFI play different roles; a proof may resort to both. The use of these rules has be coordinated. Induction prior to UFI is not a good rule of thumb and vice versa. For our example, the use of UFI is a bad starting point, as it is difficult to compute the process substitution. The root of the problem is that the recursive symbol $C^{(n)}$ is not given in terms of the indexing scheme, $k \in \{1, \ldots, n\}$, used to define *Buf*. Fortunately, induction, if successfully applied, eliminates recursive symbols. Thus, heuristics are required in order to coordinate the use of UFI and induction.

The process substitution is the key for building the output equation set in an application of UFI. However, experience indicates that computing process substitutions is an extremely difficult task. Whenever the use of UFI is suggested but the process substitution is partially solved, we let an application of the UFI method introduce a new process family to replace the process under verification. The new process family and the specification are similar in both size and structure. So the use of UFI is immediate. The proof of the new goal is used to justify the initial one. The process of replacing the program to be verified for a new process family is called a *generalisation*. Generalisation increases the search branching rate and, hence, heuristics are required in order to control its application.

The new process family is in a way incomplete. In place of some $P_i\,(i \in I)$ we put meta-variables. We use the term *meta-variable* to denote a first-order (or higher-order) unknown entity in the object-level theory. We call $\boldsymbol{\mathcal{M}}$ the set of meta-variables, and let $\mathcal{M}_0, \mathcal{M}_1, \ldots$ range over $\boldsymbol{\mathcal{M}}$. Introducing meta-variables, we can use UFI still, but at the expense of solving each equation at verification time. Thus, heuristics are required in order to control equation solving.

Summarising, automating the use of UFI prompts three major challenges: i) when and how to use UFI; ii) when and how to use generalisation; and iii) guide equation solving. These issues explain why radical measures are called for. Fortunately, as discussed below, proof planning is, at least partially, an answer to these problems.

4 Proof Planning

Proof planning [1] is a meta-level reasoning technique, developed especially as a search control engine to automate theorem proving. Proof planning works in the context of a tactical style of reasoning. It uses AI planning techniques to build large complex tactics from simpler ones, hence outlining the proof while emphasising key steps and structure.

Methods are the building-blocks of proof planning. A *method* is a high-level description of a tactic. It specifies the preconditions under which the tactic is

applicable, without actually running it, and the effects of its application. The application of a method to a given goal consists in checking the preconditions against the goal and then determining the output new goals by computing the effects.

Proof planning returns a *proof plan* (i.e., a tactic), whose execution, in the normal case of success, guarantees correctness of the final proof. Proof planning is cheaper than searching for a proof in the underlying object theory. This is both because each plan step covers a lot of ground steps, and because the method preconditions dramatically restrict the search space.

Inductive proof planning [3] is the application of proof planning to inductive theorem proving. It has been successfully applied to various domains, including software and hardware development. Inductive proof planning is characterised by the following methods: The *induction* method selects the most promising induction scheme via a process called *rippling analysis*. The base case(s) of proofs by induction are dealt with by the *elementary* and *sym_eval* methods. Elementary is a tautology checker for propositional logic and has limited knowledge of intuitionistic propositional sequents, type structures and properties of equality. Sym_eval simplifies the goal by means of exhaustive symbolic evaluation and other routine reasoning.

Similarly, the step case(s) of proofs by induction are dealt with by the *wave* and *fertilise* methods. Wave applies *rippling* [2], a heuristic that guides transformations in the induction conclusion to enable the use of an induction hypothesis. This use of an induction hypothesis, called *fertilisation* — hence the name of the method — is a crucial part of inductive theorem proving and it is handled by fertilise.

Rippling exploits the observation that an initial induction conclusion is a copy of one of the hypotheses, except for extra terms, e.g., the successor function, wrapping the induction variables. By marking such differences explicitly, rippling can attempt to place them at positions where they no longer prevent the conclusion and hypothesis from matching. Rippling is therefore an annotated term-rewriting system. It applies a special kind of rewrite rule, called a *wave-rule*, which manipulates the differences between two terms while keeping their common structure intact.

Monroy, Bundy and Green have extended inductive proof planning with a special CCS proof plan [8]. The CCS proof plan is circumscribed by the following methods: The *expansion* method transforms a concurrent form into a sum of prefixed processes. The *absorption* method gets rid of CCS terms that are redundant with respect to behaviour. The *goalsplit* method equates each process summand on one side of the equation with one on the other side, returning a collection of subgoals, each of the form $\alpha.P = \alpha.Q$. Finally, the *action* method solves equations of the form $\alpha.P = \alpha.S$ using Hennessy's theorem [7].

Monroy, Bundy and Green did not allow specifications to be recursive. In what follows, we shall show how to extend the verification plan to deal with recursion.

5 The UFI Method

The UFI method is concerned with when to apply the UFI rule. An application of the UFI method is *successful* if it introduces a minimum number of meta-variables, ideally zero. With this, it avoids unnecessary equation solving steps. The moral is that the more derivatives we associate, the more chances we have of finding a verification proof.

Concurrent forms often contain a lot of recursive symbols. Recursive symbols get in the way of an application of UFI, especially when they are not given in terms of the indexing scheme used by the specification. Then it is necessary to use induction prior to UFI in order to remove them. However induction prior to UFI is not generally a good rule of thumb.

Induction before UFI may yield a waste of effort if applied upon an index variable. This is because it would yield new goals involving P_k, for some $k \in I$, an individual of the process family. However, recall that we cannot prove that some individual satisfies a property without proving that such a property is satisfied by all. So, each goal returned by induction will need to be dealt with using UFI.

Summarising, the strategy to coordinate the use of induction and UFI is as follows. Use induction prior to UFI both if the verification problem contains recursive function symbols not given in terms of the indexing scheme, and if the selected induction variable is other than an index variable in the specification. If these conditions do not hold, UFI is applied first.

6 The Generalise Method

We now introduce a method, called *generalise*, which deals with the problem of using UFI under incomplete information. The rationale behind generalise is that we can apply UFI still, even without elaborating the details of the application. The computation of the process substitution is therefore postponed, leaving it partially defined. Subsequent planning steps are then used in order to elaborate upon the output plan, which will hopefully yield an object-level proof. The use of meta-variables to represent unknown, 'eureka' values is called *middle-out reasoning*.

Upon application, generalise builds a conditional, non-recursive process function, say $\mathcal{F}_{\mathcal{P}}$, which is put in place of the process family under verification. $\mathcal{F}_{\mathcal{P}}$ has one exceptional feature: not only does it embrace the original process, but it is also given by the same number of equations that define the specification. Thus, each equation in $\mathcal{F}_{\mathcal{P}}$ is pairwise related to one, and only one, equation in S: the application of UFI becomes immediate. $\mathcal{F}_{\mathcal{P}}$ is of course initially under-specified (and so will the output equation set). This is because all of the missing process derivatives are substituted with meta-variables, each of which it is hoped will be fixed at later planning steps. We hence change from equation verification to equation solving, which we shall discuss in §7.

Consider a verification problem, $P = S$, where the use of UFI is suggested, but where the process substitution is only partially solved. Then, to generalise the verification problem, proceed as follows:

1. Introduce a new process symbol, say $\mathcal{F}_{\mathcal{P}}$.
2. Define the new process family, so that it matches the size of S as well as its structure:

$$C_i \to H_i\{\widetilde{\mathcal{F}_{\mathcal{P}}}/\widetilde{S}\} \stackrel{\text{def}}{=} \mathcal{M}_i(P^{\to}) \text{ only if } C_i \to H_i \stackrel{\text{def}}{=} B_i \ \in \ \widetilde{S} \stackrel{\text{def}}{=} \widetilde{E}\{\widetilde{S}/\widetilde{X}\}$$

   where $P^{\to}$ denotes some P-derivative.
3. Refine the definition of $\mathcal{F}_{\mathcal{P}}$, using $P = S$ in order to fix at least one of the equation bodies, $\mathcal{M}_i$. If this process fails, so will generalise.
4. Simplify the equation system, getting rid of any redundant terms.
5. Finally, replace the original conjecture for $\mathcal{F}_{\mathcal{P}} = S$.

7 Verification = Equation Solving in CCS

Now we present a search control strategy to automatically solve the output equation set. The strategy has been designed to solve equations of the form $P = E$, where E may contain meta-variables, but P may not. Each equation in the output set is, thus, considered independently, one at a time. However, the result of any equation solving step is spread throughout the entire plan. The strategy's guiding factor is the expected behaviour that each meta-variable in E should satisfy. We call this (behavioural) constraint the *context*.

The context is nothing but a description of the process' intended behaviour, i.e., it is related to the specification. If the definition of the specification involves mutual recursion, we add the context to every subgoal so that the equation solving strategy can use it. Accordingly, we let each subgoal in the output equation set take the following schematic form:

$$\{\, P_h = E_h\{\widetilde{P}/\widetilde{X}\} \ : \ h \in I \setminus \{i\}\,\} \ldots \vdash \ P_i = E_i\{\widetilde{P}/\widetilde{X}\} \qquad (i \in I)$$

The context is a decoration. So, it is marked with a dashed box in order to make it distinguishable from the actual hypotheses.

Equation solving is applied only when the current equation subgoal is balanced. An equation is balanced iff it has the same number of process summands on each side of the equality. Equation solving involves both a process, and a meta-variable. The process and the meta-variable must be prefixed by the same action, and must appear at a different side of the equality.

The equation solving strategy is specified as follows: Solve $\mathcal{M} = P$, only if $\mathcal{M}$ and P have identical local behaviour, in symbols:

$$\forall \alpha \in \mathcal{A}ct.\ \mathcal{M} \stackrel{\alpha}{\to} \text{ if and only if } P \stackrel{\alpha}{\to}$$

While the intended behaviour of $\mathcal{M}$ is extracted from the context, the actual behaviour of P is computed using process expansion.

8 A Worked Example

We illustrate the strength of our approach by showing that full automation is achieved in proving (1). We have chosen this example because it is a challenge case study for state-of-the-art automated verification systems.

When input (1), the planner applied induction prior to UFI. This is as expected since n in $C^{(n)}$ is not an index variable in the definition of *Buf*. Afterwards, the planner solved both the base case and the step case, also fully automatically. In the step case, the planner made use of generalise when induction returned the goal below:

$$\forall n\!:\!\text{nat}.\ n \neq 0 \rightarrow C^{\frown} Buf_{\langle n,0\rangle} = Buf_{\langle s(n),0\rangle} \tag{3}$$

Generalise then automatically tackled the new problem outputting the formula $\forall n : \text{nat}.\ n \neq 0 \rightarrow \mathcal{F}_{\mathcal{P}\langle s(n),0\rangle} = Buf_{\langle s(n),0\rangle}$, where $\mathcal{F}_{\mathcal{P}}$ was initially given by:

$$\begin{aligned}
&s(n) \neq 0 \rightarrow \mathcal{F}_{\mathcal{P}\langle s(n),0\rangle} \stackrel{\text{def}}{=} C^{\frown} Buf_{\langle n,0\rangle}\\
&s(n) > j \rightarrow \mathcal{F}_{\mathcal{P}\langle s(n),j\rangle} \stackrel{\text{def}}{=} \mathcal{M}_1(\mathcal{M}_{11}(C,D), Buf_{\langle n,\mathcal{M}_{12}(j)\rangle})\\
&s(n) = j \rightarrow \mathcal{F}_{\mathcal{P}\langle s(n),j\rangle} \stackrel{\text{def}}{=} \mathcal{M}_2(\mathcal{M}_{21}(C,D), Buf_{\langle n,\mathcal{M}_{22}(j)\rangle})
\end{aligned}$$

Note that the $\mathcal{M}$s are all meta-variables. Also note that $\mathcal{M}_{11}(C,D)$ sufficed to represent the state space of C: $\{C,D\}$, and similarly for $\mathcal{M}_{21}(C,D)$ and $Buf_{\langle n,\mathcal{M}_{22}(j)\rangle}$.

With the revised conjecture, the use of UFI was immediate, yielding:

$$\begin{aligned}
&\vdash s(n) \neq 0 \ \rightarrow\ PF_{\langle s(n),0\rangle} = in.PF_{\langle s(n),s(0)\rangle}\\
&\vdash s(n) > k \ \rightarrow\ PF_{\langle s(n),s(k)\rangle} = in.PF_{\langle s(n),s(s(k))\rangle} + \overline{out}.PF_{\langle s(n),k\rangle}\\
&\vdash s(n) = k \ \rightarrow\ PF_{\langle s(n),s(k)\rangle} = \overline{out}.PF_{\langle s(n),k\rangle}
\end{aligned} \tag{4}$$

Each goal in the output equation was tackled successfully. For example, when the equation solving strategy had been already used to successfully fix $\mathcal{M}_1(\mathcal{M}_{11}(C,D), Buf_{\langle n,\mathcal{M}_{12}(j)\rangle})$ to $C^{\frown} Buf_{\langle n,j\rangle}$, the current definition of $\mathcal{F}_{\mathcal{P}}$ was as above except for the second case: $s(n) > j \rightarrow \mathcal{F}_{\mathcal{P}\langle s(n),j\rangle} \stackrel{\text{def}}{=} C^{\frown} Buf_{\langle n,j\rangle}$. The working subgoal then was (4):

$$\begin{aligned}
&s(n) \neq 0 \rightarrow\ \ \mathcal{F}_{\mathcal{P}\langle s(n),0\rangle} = in.\mathcal{F}_{\mathcal{P}\langle s(n),s(0)\rangle}\\
&s(n) > k \rightarrow \mathcal{F}_{\mathcal{P}\langle s(n),s(k)\rangle} = in.\mathcal{F}_{\mathcal{P}\langle s(n),s(s(k))\rangle} + \overline{out}.\mathcal{F}_{\mathcal{P}\langle s(n),k\rangle}\\
&s(n) = k \rightarrow \mathcal{F}_{\mathcal{P}\langle s(n),s(k)\rangle} = \overline{out}.\mathcal{F}_{\mathcal{P}\langle s(n),k\rangle}\\
&n : \text{nat},\ n \neq 0\\
&\vdash\\
&s(n) > s(k)\ \rightarrow\ C^{\frown} Buf_{\langle n,s(k)\rangle} = in.\mathcal{F}_{\mathcal{P}\langle s(n),s(s(k))\rangle} + \overline{out}.\mathcal{F}_{\mathcal{P}\langle s(n),k\rangle}
\end{aligned}$$

With this goal, a further application of equation solving was required. It occurred after the application of casesplit upon the partition: $[n > s(k), n = s(k)]$. The

interesting case is $n = s(k)$, where the planner applied expansion, together with sym_eval, leaving:

$$\ldots \vdash \; in.(D^\frown Buf_{\langle n,s(k)\rangle}) + \overline{out}.(C^\frown Buf_{\langle n,k\rangle}) = \\ in.\mathcal{M}_2(\mathcal{M}_{21}(C,D), Buf_{\langle n,\mathcal{M}_{22}(s(k))\rangle}) + \overline{out}.(C^\frown Buf_{\langle n,k\rangle})$$

Then, second-order matching suggested the following substitution:

$$\mathcal{M}_2 \mapsto \frown, \;\; \mathcal{M}_{21} \mapsto \mathrm{inr}(C,D), \;\; \mathcal{M}_{22} \mapsto \lambda x.x$$

The solution met the constraint imposed by the context, because, by expansion, $D^\frown Buf_{\langle n,s(k)\rangle}$ equals $\overline{out}.(D^\frown Buf_{\langle n,k\rangle})$. So, it was accepted, yielding a refinement on the third defining equation of $\mathcal{F}_\mathcal{P}$: $s(n) = j \to \mathcal{F}_{\mathcal{P}\,\langle s(n),j\rangle} \stackrel{\text{def}}{=} D^\frown Buf_{\langle n,j\rangle}$, which is just as required.

9 Results, Related Work, and Conclusions

Table 1 gives some of the example systems with which we tested our proof plan. PT stands for the total elapsed planning time, given in seconds. Space constraints do not allow us to provide a full description of each verification problem. These example verification problems are all taken from [7]. They all involve the use of generalise, UFI and equation solving. The success rate of the proof plan was

Table 1. Some example verification conjectures

Conjecture	Description	PT (sec)
n-bit counter	A chain of n cells, each acting as a 1-bit counter, implements a counter of size n	57
n-bit sorter	A chain of n cells, each acting as a filter implements a bubble-sort algorithm for a string of elements	614
semaphore array	A set containing n 1-bit semaphores implements a semaphore of size n	992
cycler array	A collection of cyclers computes the sequence $a_1.\cdots.a_n.a_1\cdots$ indefinitely	411

83%, with an average total elapsed planning time of 750 seconds, and standard deviation of 345. The test was run on a Solbourne 6/702, dual processor, 50MHz, SuperSPARC machine with 128 Mb of RAM. The operating system, Solbourne OS/MP, is an optimised symmetric multi-processing clone of SunOS 4.1. The full test set, including the planner, are available upon request, by sending electronic-mail to the first author.

CCS has been mechanised in several proof checkers. For example, Cleaveland and Panangaden describe an implementation of CCS in Nuprl [4]. Cleaveland

and Panangaden did not strive for automation; instead, they were interested in showing the suitability of type theory for reasoning about concurrency in general. Also Nesi implemented CCS in HOL [9]. But Nesi's motivation was somewhat different: to show the suitability of (induction oriented) proof checkers for reasoning about parameterised systems. The tool was the first to accommodate the analysis of parameterised systems, but did not improve upon the degree of automation. Lin reported an implementation of a CCS like value-passing process algebra in VPAM [6], a generic proof checker. These verification frameworks are highly interactive, requiring at each step the user to select the tactic to be applied. By contrast, our verification planner is fully automatic, accommodating parameterised, infinite-state systems. However, unlike Lin, we do not accommodate truly value-passing systems, relying upon infinite Summation to simulate it.

Concluding, the Verification planner handles the search control problems prompted by the use of UFI more than satisfactorily. We have planned proofs of conjectures that previously required human interaction. Full automation is an unattainable ideal, but we should nevertheless strive towards it. We intend to apply the verification planner to the verification of larger, industrial strength examples, and see what extensions are required.

References

1. A. Bundy. The use of explicit plans to guide inductive proofs. In R. Lusk and R. Overbeek, editors, *9th Conference on Automated Deduction*, pages 111–120. Springer-Verlag, 1988.
2. A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, and A. Smaill. Rippling: A heuristic for guiding inductive proofs. *Artificial Intelligence*, 62:185–253, 1993.
3. A. Bundy, F. van Harmelen, J. Hesketh, and A. Smaill. Experiments with proof plans for induction. *Journal of Automated Reasoning*, 7:303–324, 1991.
4. R. Cleaveland and P. Panangaden. Type Theory and Concurrency. *International Journal of Parallel Programming*, 17(2):153–206, 1988.
5. ISO. *Information processing systems - Open Systems Interconnection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour*. ISO 8807, 1989.
6. H. Lin. A Verification Tool for Value-Passing Processes. In *Proceedings of* 13^{th} *International Symposium on Protocol Specification, Testing and Verification*, IFIP Transactions. North-Holland, 1993.
7. R. Milner. *Communication and Concurrency*. Prentice Hall, London, 1989.
8. R. Monroy, A. Bundy, and I. Green. Planning Equational Verification in CCS. In D. Redmiles and B. Nuseibeh, editors, *13th Conference on Automated Software Engineering, ASE'98*, pages 43–52, Hawaii, USA, 1998. IEEE Computer Society Press. Candidate to best paper award.
9. M. Nesi. Mechanizing a proof by induction of process algebra specifications in higher-order logic. In K. G. Larsen and S. A., editors, *Proceedings of the 3rd International Workshop in Computer Aided Verification (CAV'91)*. Springer Verlag, 1992. Lecture Notes in Computer Science No. 575.

Knowledge Representation Using High-Level Non-monotonic Reasoning

Mauricio Osorio[1], Juan Carlos Nieves[1], Fernando Zacarias[2], and Erika Saucedo[3]

[1] Universidad de las Americas,
Departamento de Ingenieria en Sistemas Computacionales,
Sta. Catarina Martir, Cholula
72820 Puebla, Mexico
josorio@mail.udlap.mx
[2] Benemerita Universidad Autonoma de Puebla,
Facultad de ciencias de la computación,
75570 Puebla, Mexico.
[3] Instituto Tecnológico y de Estudios Superiores de Monterrey, CEM,
Departamento de Ingenieria y Ciencias
Atizapan de Zaragoza, Edo. de Mexico.
esaucedo@campus.cem.itesm.mx

Abstract. We introduce the new paradigm of High-Level Non-Monotonic reasoning (HLNM). This paradigm is the consolidation of our recent results on disjunctions, sets, explicit and implicit negation, and partial-order clauses. We show how these concepts are integrated in a natural way into the standard logic programming framework. For this purpose, we present several well known examples from the literature that motivate the need of this new paradigm. Finally, we define a declarative semantics for HLNM reasoning.

1 Introduction

Knowledge based systems can be modeled adequately using non-monotonic logic, because it provides formal methods that enable intelligent systems to operate properly when they are faced with incomplete or changing information. There exist close interconnections between logic programming and non-monotonic reasoning. Actually, negation as failure, the interpretation of negation in logic programs as the failure to prove a ground atomic formula is a prothotypical example of non-monotonic behavior.

In this paper, we introduce the new paradigm High-Level Non-Monotonic reasoning, which supports the well-known formalism of definite logic programs by allowing for disjunctions in rule heads, general formulas in rule bodies, explicit and implicit negation, set notation $\{H|T\}$ like Prolog's list notation, and function atoms $f(t) = e$ or $f(t) \geq e$ for formulating conditions on functions like $intersect(\{X|\_\}, \{X|\_\}) \geq \{X\}$. Thus, High-Level Non-Monotonic reasoning integrates functional programming aspects, since functions may be defined by

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 13–24, 2000.

logical rules. One main contribution of our paper is the definition of a semantics integrating all the above mentioned concepts.

The interest in allowing two kinds of negations (default negation and explicit negation) is considered elsewhere in the literature, see [4,3,7,18]. It has been shown in [3] how they can be used to represent defeasible reasoning. In [28] we discuss an application to the Botanic filed. Based on the work in [18] we also claim that HLNM reasoning is better suited for legal knowledge representation than normal logic programming, due to the presence, in HLNM, of both explicit and default negation.

With respect to default negation, it is well known that the stable semantics and the well founded semantics (WFS) are the two most acceptable definitions of this form of negation in Logic Programming. Our use of default negation is different than in previous approaches. We allow the combination of the idea of negation as failure with reasoning by cases to obtain a powerful use of negation. Preliminary research on this direction is presented in [16,10].

Disjunctive logic programming is one of the most expressive declarative logic programming language [8]. It is more natural to use since it permits direct translation of disjunctive statements from natural language and from informal specifications. The additional expressive power of disjunctive logic programs significantly simplifies the problem of translation of non-monotonic formalisms into logic programs, and, consequently, facilitates using logic programming as an inference engine for non-monotonic reasoning. A novel contribution of our paper is a translation T from HLNM programs P to disjunctive logic programs (with default negation) P'.

Let us motivate the need for partial-order clauses. In a recent meeting organized by Krzyststof Apt (president of the Association of Logic Programming), whose aim was to get a clear picture of the current activities in Logic Programming and its role in the coming years, Gelfond pointed out: "We need to find elegant ways for accomplishing simple tasks, e.g. counting the numbers of elements in the database satisfying some property P. At the moment we have neither clear semantics nor efficient implementation of *setof* of other aggregates used for this type of problems" [9]. We propose partial-order clauses as an answer to this issue. The evolution of partial-order clauses is shown in the papers [25,26,23,22,15,29]. Related work on this topic is found in the following papers: [2,20,13,12,31,1,19,30]. A central concept in most of the works that include sets is one which we call *collect-all*. The idea is that a program needs only to define explicitly what are the members of the set and, by assuming a *closed world*, the set becomes totally defined. Moreover, all the just mentioned works, have defined a precise semantics for restricted classes of programs and they agree on the common subclasses. Also, a common assumption is to consider only finite sets. According to our knowledge, our approach generalizes all the work done so far about this topic.

HLNM programs also allow general formulas as the body of a clause. In [17,21] it is found motivation on this issue.

A preliminary research on this topic is presented in [29]. The current paper presents the full language and a more refined semantics.

Our paper is structured as follows: In section 2, we introduce our new paradigm High-Level Non-Monotonic reasoning, giving its grammatical definition and motivation. In section 3, we define the declarative semantics of our language. Finally, in our section we present our conclusions.

2 High-Level Non-monotonic Programs

We will now give a brief informal introduction to HLNM programs to give an intuition of the style of programming in this language and also to provide some context for discussing the semantic issues it raises. HLL clauses have the following form:

$$
\begin{array}{ll}
rule & ::= head \leftarrow body. \mid head. \\
head & ::= literal_s \mid literal_s \ ; \ head \\
literal_s & ::= literal_e \mid POC \\
POC & ::= Fu(list\_terms) \geq Fu(argument_f) \mid Fu(list\_terms) \geq term \\
body & ::= literal_b \mid (body \ Ob \ body) \mid (Q \ Var \ body) \mid (\neg body) \\
literal_b & ::= atom_f \mid literal_e \\
literal_e & ::= -atom \mid atom \\
atom & ::= Pr(list\_terms) \\
atom_f & ::= Fu(list\_terms) = term \\
arguments_f & ::= argument_f \mid argument_f, \ arguments_f \\
argument_f & ::= f(arguments_f) \mid terms \\
list\_terms & ::= term \mid term, \ list\_terms \\
term & ::= Const \mid Var \mid CT(list\_terms)
\end{array}
$$

where Ob, Q, Pr, Fu, CT, and $Cons$ are nonterminal symbols defined as follows: $Ob \in \{\wedge, ;, \rightarrow\}$, $Q \in \{\exists, \forall\}$, Pr is a user-defined predicate symbol, Fu is any functional symbol, CT is any contructor symbol, and $Cons$ is any constant symbol (i.e. a constructor of arity 0). We use ";" to denote the disjunctive logical connective. We also use "," to denote the conjunctive logical connective $\wedge$. We use $-$ and $\neg$ to denote explicit and default negation respectively. A HLNM program is a set of program rules (also called clauses).

We adopt four constructors: *cons* (of arity 2), *nil* (of arity 0), *scons* (of arity 2), and *empty* (of arity 0). The two constructors *cons* and *nil* are used to represent lists as usual in logic programming. Moreover, we prefer the notation $[X|Y]$ instead of $cons(X, Y)$ and [] instead of *nil*. To represent finite sets, we use the two constructors *scons* and *empty*. Again, we prefer to use the more suggestible notation $\{x \backslash t\}$ and ϕ. The notation $\{x \backslash t\}$ refers to a set in which x is one element and t is the remainder of the set. We permit as syntactic sugar $\{expr\}$ to stand for $\{expr \backslash \phi\}$, and $\{e_1, e_2, \ldots, e_k\}$, where all e_i are distinct, to stand for $\{e_1 \backslash \{e_2 \backslash \ldots \{e_k \backslash \phi\}\}\}$. To illustrate the set constructor, matching $\{\mathtt{X} \backslash \mathtt{T}\}$ against a ground set $\{\mathtt{a, \ b, \ c}\}$ yields three different substitutions, $\{\mathtt{X} \leftarrow \mathtt{a}, \mathtt{T} \leftarrow \{\mathtt{b}, \mathtt{c}\}\}$, $\{\mathtt{X} \leftarrow \mathtt{b}, \mathtt{T} \leftarrow \{\mathtt{a}, \mathtt{c}\}\}$, and $\{\mathtt{X} \leftarrow \mathtt{c}, \mathtt{T} \leftarrow \{\mathtt{a}, \mathtt{b}\}\}$. One should contrast $\{\mathtt{X} \backslash \mathtt{T}\}$ from $\{\mathtt{X}\} \cup \mathtt{T}$.

The 'logical' view of logic programming is that a program is a theory and computation is logical inference. In the classical approach to logical programming, the inference mechanism is faithful to *logical consequence* in classical first-order logic. This paper adopted the *canonical model* approach, where the inference mechanism is faithful to truth in the "intended" model of the program.

2.1 Negation in Logic Programming.

Default negation has proved to be quite useful in many domains and applications, however, this in not the unique form of negation which is needed for non-monotonic formalisms. Explicit negation has many characteristics which makes it a good candidate to represent non-monotonic reasoning. It can occur in the head of a clause and so it allows us to conclude negatively. Explicit Negation treats negative information and positive information in a simetric form, that is to say, it does not give any kind of preference. However, it allows the representation of exceptions and preference rules. We show how can we represent defeasible reasoning with default and explicit negation. In particular, we discuss the representation of exceptions and preference rules. These ideas are not ours, but taken from [3]. The notion of exception may be expressed in three different ways:

Exceptions to predicates. We express that the rule $angiosperm(X) \longleftarrow tree(X)$ applies whenever possible but can be defeated by exceptions using the rule:

$\texttt{angiosperm(X)} \longleftarrow \texttt{tree(X)}, \neg\ \texttt{ab(X)}$

If there is a tree `a` which is known that is not an angiosperm we may express it by $\neg$`angiosperm(a)`. In this case $\neg$`angiosperm(a)` establishes an exception to the conclusion predicate of the defeasible rule.

Exceptions to rules. A different way to express that a given element is some exception is to say that a given rule must not be applicable to the element. If, for instance, element `a` is an exception to the rule *trees are angiosperms*, we express it as `ab(a)`. In general, we may want to express that a given `X` is abnormal under certain conditions. This is the case where we want to express *Pines are abnormal* to the rule about `angiosperms` given above. We write this as follows:

$\texttt{ab(X)} \longleftarrow \texttt{pine(X)}$

Exceptions to exceptions. In general we may extend the notion of exceptioned rules to exception rules themselves, i.e. exception rules may be defeasibles. This will allow us to express an exception to an exception rule.

Preference rules. We may express now preference between two rules, stating that if one rule may be used, that constitutes an exception to the use of the other rule:

$\texttt{angiosperm(X)} \longleftarrow \texttt{tree(X)}, \neg\ \texttt{ab}_1\texttt{(X)}$

$\texttt{- angiosperm(X)} \longleftarrow \texttt{pine(X)}, \neg\ \texttt{ab}_2\texttt{(X)}$

$\texttt{tree(X)} \longleftarrow \texttt{pine(X)}$

In some cases we want to apply the most specific information; above, there should be (since a pine is a specific kind of tree) an explicit preference of the rule about non-angiosperm pines over the rule about angiosperm trees.

$\texttt{ab}_1\texttt{(X)} \longleftarrow \texttt{pino(X)}, \neg\ \texttt{ab}_2\texttt{(X)}$

Botanic Field. We have studied a small fragment part of the Botanic field (the Mycota Kingdom), where we found that is natural to use exceptions and preference rules. Our information was taken from the *Britanic Encyclopedia OnLine* and our results are presented in [28].

2.2 Disjunctions

Disjunctive reasoning is more expressive and natural to use since it permits direct translation of disjunctive statements from natural language and from informal specifications. The additional expressive power of disjunctive logic programs significantly simplifies the problem of it translation of non-monotonic formalisms into logic programs, and consequently, facilitates using logic programming as an inference engine for non-monotonic reasoning.
The following is a well known example that can not be handled adequately by Circumscription.

Example 1 (Poole's Broken arm, [7]).

```
left-use(X) ← ¬ ab(left,X).
ab(left,X) ← left-brok(X).
right-use(X) ← ¬ ab(right,X).
ab(right,X) ← right-brok(X).
left-brok(fred) ; right-brok(fred) ←.
make-cheque(X) ← left-use(X).
make-cheque(X) ← right-use(X).
disabled(X) ← left-brok(X), right-brok(X).
```

The well known semantics D-WFS and DSTABLE derive that Fred is not disabled (see [7]). We get the same answer in our approach. Moreover, DSTABLE (but not D-WFS) derives that Fred can make out a cheque. We get also this result in our approach.

2.3 Partial-Order Clauses

As we have said, Gelfond pointed out: "We need to find elegant ways for accomplishing simple tasks, e.g. counting the numbers of elements in the database satisfying some property P. At the moment we have neither clear semantics nor efficient implementation of *setof* of other aggregates used for this type of problems" [9]. We propose to use partial-order clauses to solve this problem. The first author of this paper has done an extensive research on this topic in [25,26,23,22,15,29] and we have strong evidence to make us think that our results in this paper will help to obtain a final answer.

Modeling Setof. The original motivation of partial-order clauses was to include a kind of setof operator. There are two mayor problems with the setof operator in PROLOG. First, it has no formal definition and second, it is very weak. See [23].

Example 2. Consider that we want to obtain the set of all students that are taking both the cs101 class and cs202 class. We write a clause in Prolog that do the given function as follow:

$\texttt{both(X)} \ :- \ \texttt{setof(W,(cs101(W),cs202(W)),X)}$

while in our paragigm this represented by:

$\texttt{both(X)} \ \geq \ \{\texttt{X}\} \leftarrow \texttt{cs101(W)},\texttt{cs202(W)}$

We note that our paradigm is not only a change of notation, but we define a more powerful notion of 'setof' than PROLOG. The reader is invited to see an example of this nature in [29].

General Domains. We generalize our approach of partial-orders to different domains and not just sets. The following example illustrates this idea.

Example 3 (Min-Max Program1 [13]).

$\texttt{p(X)} \leq \texttt{X1} \ \leftarrow \texttt{final(X,X1).}$
$\texttt{p(X)} \leq \texttt{X2} \ \leftarrow \texttt{edge2(X,Y), q(Y)=X2.}$
$\texttt{q(X)} \geq \texttt{X3} \ \leftarrow \texttt{final(X,X3).}$
$\texttt{q(X)} \geq \texttt{X4} \ \leftarrow \texttt{edge1(X,Y), p(Y)=X4.}$

This program models a two-player game defined on a bipartite graph represented by predicates `edge1` and `edge2`. The function `p` obtains the minimum value, while `q` obtains the maximum value. The program is considered non-monotonic due to the interaction of $\geq$ with $\leq$. Consider the following extensional database:

```
final(d,0).  edge1(a,b).  edge2(b,d).
final(e,1).  edge1(a,c).  edge2(b,e).
final(f,-1).              edge2(c,f).
final(g,0).               edge2(c,g).
```

In our approach we obtain the intended model, where `p(a)=0`. Our operational semantics with pruning (introduced in [27]) behaves very much as the well known alpha-beta procedure. Other interesting examples about partial-order are presented in [23].

It is important to observe, as will see in the following section, that we have a formal definition of the semantics of partial-order clauses.

3 Declarative Semantics

We first present our definition of the declarative semantics for propositional disjunctive programs. Then we explain how we generalize our results to the general framework of HLNM programs.

3.1 Declarative Semantics of Propositional Disjunctive Programs

A *signature* is any finite set. An *atom* is any element of a signature. A *positive literal* is an atom. A *negative literal* is a formula of the form $\neg a$, where a is an atom. A *literal* is a positive literal or a negative literal. Two literals are of the same sign if both are positive or both are negative. Propositional disjunctive clauses are of the form:

$A_1; \ldots; A_n \leftarrow L_1, \ldots L_m$

where $n \geq 1, m \geq 0$, every A_i is an atom and every L_i is a literal. When $m = 0$, we say that the clause is a fact. When $n = 1$ the clause is called normal. A normal program consists of just normal clauses. Traditional logic programming is based on this kind of programs. When $m = 0$ and $n = 1$ the clause is called a normal fact. We also would like to denote a disjunctive clause by $C{:=}\ \mathcal{A} \leftarrow \mathcal{B}^+, \neg\mathcal{B}^-$, where $\mathcal{A}$ denotes the set of atoms in the head of the clause (namely $\{A_1, \ldots A_n\}$;), $\mathcal{B}^+$ denotes the set of positive atoms in the body, and $\mathcal{B}^-$ denotes the set of negative atoms in the body. A pure disjunction a disjunction of literals of the same sign.

The stable semantics and the well founded semantics are the two most well known definitions of semantics for Logic Programs. Our proposed semantics is different than in previous aproaches and more powerful even for normal programs. We allow the combination of the idea of negation as failure with reasoning by cases to obtain a powerful use of negation.

We now provide our formal definition of our declarative semantics.
We start with some definitions. Given a program P, we define $HEAD(P) := \bigcup_{\mathcal{A} \leftarrow \mathcal{B}^+, \neg\mathcal{B}^- \in P} \mathcal{A}$. It will be useful to map a disjunctive program to a normal program. Given a clause $C{:=}\ \mathcal{A} \leftarrow \mathcal{B}^+, \neg\mathcal{B}^-$, we write *dis-nor(C)* to denote the set of normal clauses:

$\{a \leftarrow \mathcal{B}^+, \neg(\mathcal{B}^- \cup (\mathcal{A} \setminus \{a\}) | a \in \mathcal{A})\}$.

We extend this definition to programs as follows. If P is a program, let *dis-nor(P)* denotes the normal program: $\bigcup_{C \in P} dis - nor(C)$.

We first discuss the notion of a supported model for a disjunctive program. It generalizes the notion of a supported model for normal programs (which in turns is equivalent to a model of the Clark's completion of a program).

Definition 1 (Supported model).
Let P be a program. A supported model M of P is model of $dis - nor(P)$ such that for every atom a that occurs in P and is true in M, there exists a clause C in $dis - nor(P)$ such that a is the head of C and the body of C is true in M.

We now discuss the notion of a D-WFS partial model. The key concept of this approach is the idea of a *transformation* rule. We adopt the transformation rules introduced in [5,7], which are: RED$^+$, RED$^-$, GPPE, TAUT, SUB. We define $\mathcal{CS}_1$ to be the rewriting system which contains the just mentioned rules. It is known that this system is confluent and terminating, see [6].

Definition 2 (D-WFS partial model). *Let P be a program and P' its normal form, i.e. P' is the program obtained after reducing P by $\mathcal{CS}_1$. The D-WFS*

partial model of P is defined as: $\{a|$ *a is a normal fact of* $P'\ \} \cup \{\neg a|$ *a is an atom in the language of P that does not occurs in* $P'\ \}$.

We can now present the main definition of this section:

Definition 3. *Let P be a program. We define a DWFS-DCOMP model as a two-valued minimal supported model that extends D-WFS(P). Any such extension is a model that agrees with the true/false assigments given by D-WFS(P). The scenario DWFS-DCOMP semantics of P is the set of DWFS-DCOMP models of P. The sceptical DWFS-DCOMP semantics for P is the set of pure disjunctions that are true in every DWFS-DCOMP model of P. If no such model exists then we say that the program is inconsistent.*

In the above definition we stress the importance of considering minimal models ([21]), since they are critical in logic programming. In this paper our concern is only in the sceptical semantics.

3.2 Moving to Predicates and Adding the Set Constructor

We will work with Herbrand interpretations, where the Herbrand Universe of a program P consists only of ground terms, and is referred to as U_P. The Herbrand Base B_P of a program P consists of ground atoms as usual.

The generalization of the function $dis - nor$ to predicate logic is straightforward. Given this , we need a definition of supported model for predicate normal programs. This has been done in [21]. D-WFS for predicate programs is presented in full detail in [11]. We can therefore define DWFS-DCOMP models with no problem.

With respect to the set constructor we proceed as follows. We continue working with Herbrand interpretations. But due to the equational theories for constructors, the predicate = defines an equivalence relation over the Herbrand Universe. But, we can always *contract* a model to a so-called normal model where = defines only an identity relation [24] as follows: Take the domain D' of I to be the set of equivalence classes determined by = in the domain U_P of I. Then use Herbrand $\equiv$-interpretations, where $\equiv$ denotes that the domain is a quotient structure. We then should refer to elements in D' by $[t]$, i.e. the equivalence class of the element t, but in order to make the text more readable, we will refer to the $[t]$ elements just as t, keeping in mind that formally we are working with the equivalence classes of t. These details are explained in [14].

3.3 Adding Partial-Order Clauses

The first step is to obtain the flattened form [14] of every partial-order clause. Consider the following example:

`f(X)` $\geq$ `g(h(X)) ; f(Z)` $\geq$ `{m(Z)}` $\leftarrow$ `l(X).`

then its flattened form is:

`(f(X)` $\geq$ `X1` $\leftarrow$ `h(X)=X2, g(X2)=X1 ; f(Z)` $\geq$ `{ Z1 }` $\leftarrow$ `m(Z)= Z1)` $\leftarrow$ `l(X).`

The second step is to transform this formula to a disjunctive clause. With this same example, we get:

$\texttt{f(X)} \geq \texttt{X1; f(Z)} \geq \{\ \texttt{Z1}\ \} \leftarrow \texttt{h(X)=X2, g(X2)=X1, m(Z)=Z1, l(X).}$

As the third step we translate the clause to its relational form. Again, using this example, we get:

$\texttt{f}_{\geq}\texttt{(X,X1); f}_{\geq}\texttt{(Z,\{Z1\})} \leftarrow \texttt{h}_{=}\texttt{(X,X2), g}_{=}\texttt{(X2,X1), m}_{=}\texttt{(Z,Z1), l(X).}$

These steps can easily be defined for the general case. We suggest the reader to see [26] to check the details.

The fourth step is to add axioms that related the predicate symbols $\texttt{f}_{=}$ with $\texttt{f}_{\geq}$ for each functional symbol f. Let us consider again our example. The axioms for f in this case are as follows:

(1) $\texttt{f}_{=}\texttt{(Z, S)} \leftarrow \texttt{f}_{\geq}\texttt{ (Z, S)},\ \neg\, \texttt{f}_{>}\texttt{(Z, S)}$

(2) $\texttt{f}_{>}\texttt{(Z, S)} \leftarrow \texttt{f}_{\geq}\texttt{(Z,S1)},\ \texttt{S1 > S}$

(3) $\texttt{f}_{\geq}\texttt{(Z, S)} \leftarrow \texttt{f}_{\geq}\texttt{(Z,S1)},\ \texttt{S1 > S}$

(4) $\texttt{f}_{\geq}\texttt{ (Z, }\bot\texttt{)}$

(5) $\texttt{f}_{\geq}\texttt{(Z,C)} \leftarrow \texttt{f}_{\geq}\texttt{(Z,C}_1\texttt{)},\ \texttt{f}_{\geq}\texttt{(Z,C}_2\texttt{)},\ \ \texttt{lub(C}_1\texttt{,C}_2\texttt{,C).}$

We understand that `S1 > S` means that $\texttt{S1} \geq \texttt{S}$ and $\texttt{S1} \neq \texttt{S}$. $\bot$ is a constant symbol used to represent the bottom element of the lattice and $\texttt{lub}(C_1, C_2, C)$ interprets that C is the least upper bound of C_1 and C_2. The first two clauses are the same (modulo notation) as in definition 4.2 in [32]. Clause (5) is not useful for total-order domains. It is easy to see that symmetric definitions can be provided for $f_{\leq}$ symbols.

3.4 Allowing a General Formula as the Body Clause

Here, we adopt the approach suggested by Lloyd in his well known book [21]. The idea is to apply an algorithm that transform a clause (with a general formula as the body) into one or more clauses with simple bodies (i.e. conjunction of literals). Space limitations disallow us to explain the algorithm, instead, we work out an example and let the interested reader to check the details in the above mentioned reference.

$\texttt{rp}(\{\texttt{X}\backslash\_\}) \geq \{\texttt{X}\} \leftarrow \forall \texttt{Y}(\texttt{d}(\texttt{X},\texttt{Y}) \rightarrow \texttt{ontime}(\texttt{X},\texttt{Y}))$

If we use the rules of [21] the translation becomes:

$\texttt{rp}(\{\texttt{X}\backslash\_\}) \geq \{\texttt{X}\} \leftarrow \neg\texttt{c}(\texttt{X})$

$\texttt{c}(\texttt{X}) \leftarrow \texttt{d}(\texttt{X},\texttt{Y}), \neg\texttt{ontime}(\texttt{X},\texttt{Y}).$

where c is new predicate symbol.

3.5 Explicit Negation

We show how to reduce programs with explicit negation to programs without it. The idea is originally considered in [4] but we have also explored it in [28]. Let T be the program that includes explicit negation. Let P be the program T plus the following set of clauses:

$-p(X_1, \ldots, X_n) \leftarrow \neg p(X_1, \ldots X_n)$

for all predicate symbols p from the language of T. For any predicate symbol p ocurring in P, let p' be a new predicate symbol of the same arity. The atom $p'(t_1, \ldots p_n)$ will be called the *positive form* of the negative $literal_e$ $-p(t_1, \ldots p_n)$. Every positive $literal_e$ is, by definition its own form. Let P' the program that we obtain form P by replacing every $literal_e$ by its positive form.

There is a simple way of evaluating queries in T. To obtain an answer for p run queries p and p' on P'. If the answer to p is yes then we say that T derives p. If the answer to p' is yes then we say that T does not derives p.

4 Conclusions

We presented our notion of HLNM program. It includes standard logic programming plus disjunctions, partial-order clauses, two kinds of negation and general body formulas. For this purpose, we had to define a new declarative semantics (an hybrid of D-WFS and supported models) that is suitable to express the above mentioned notions. To our knowledge, this is the first proposal that extends Logic Programming that far. We presented several and different interesting problems considered in the literature. HLNN programming defines the intended meaning of each of them. Our paradigm combines several ideas from several authors and our own earlier work and it seems promising.

The bad news is that the operational semantics of the full language is not computable. However we can identify large and interesting fragments of the language that are computable. Our main research on this direction is presented in [23,22].

References

1. E. Pontelli A. Dovier, E. G. Omodeo and G. Rossi. {log}: A logic programming language with finite sets. In *Proc. 8th International Conference of Logic Programming*, pages 111–124, Paris, June 1991. 1991.
2. S. Abiteboul and S. Grumbach. A rule-based language with functions and sets. *ACM Transactions on Database Systems*, 16(1):1–30, 1991.
3. Jose Julio Alferes and Luiz Moniz Pereira, editors. *Reasoning with Logic Programming*, LNAI 1111, Berlin, 1996. Springer.
4. Chitta Baral and Michael Gelfond. Logic Programming and Knowlege Representation. *Journal of Logic Programming*, 19-20:73–148, 1994.
5. Stefan Brass and Jürgen Dix. Characterizations of the Disjunctive Stable Semantics by Partial Evaluation. *Journal of Logic Programming*, 32(3):207–228, 1997. (Extended abstract appeared in: Characterizations of the Stable Semantics by Partial Evaluation *LPNMR, Proceedings of the Third International Conference, Kentucky*, pages 85–98, 1995. LNCS 928, Springer.).
6. Stefan Brass and Jürgen Dix. Characterizations of the Disjunctive Well-founded Semantics: Confluent Calculi and Iterated GCWA. *Journal of Automated Reasoning*, 20(1):143–165, 1998. (Extended abstract appeared in: Characterizing D-WFS: Confluence and Iterated GCWA. *Logics in Artificial Intelligence, JELIA '96*, pages 268–283, 1996. Springer, LNCS 1126.).

7. Gerhard Brewka and Jürgen Dix. Knowledge representation with logic programs. Technical report, Tutorial Notes of the 12th European Conference on Artificial Intelligence (ECAI '96), 1996. Also appeared as Technical Report 15/96, Dept. of CS of the University of Koblenz-Landau. Will appear as Chapter 6 in *Handbook of Philosophical Logic*, 2nd edition (1998), Volume 6, Methodologies.
8. Baral C. and Son T.C. Formalizing sensing actions: a transition function based approach. In *Cognitive Robotics Workshop of AAAI Fall Symposium*, pages 13–20, 1998.
9. Jürgen Dix. The Logic Programming Paradigm. *AI Communications*, Vol. 11, No. 3:39–43, 1998. Short version in Newsletter of ALP, Vol. 11(3), 1998, pages 10–14.
10. Jürgen Dix, Mauricio Osorio, and Claudia Zepeda. A General Theory of Confluent Rewriting Systems for Logic Programming and its Applications. *Annals of Pure and Applied Logic*, submitted, 2000.
11. Jürgen Dix and Frieder Stolzenburg. A Framework to incorporate Nonmonotonic Reasoning into Constraint Logic Programming. *Journal of Logic Programming*, 37(1,2,3):47—76, 1998. Special Issue on *Constraint Logic Programming*, Guest Editors: Kim Marriott and Peter Stuckey.
12. S. Greco G. Ganguly and C. Zaniolo. Minimum and maximum predicates in logic programs. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 154–16. 1991.
13. A. Van Gelder. The well-founded semantics of aggregation. In *Proc. ACM 11th Principles of Database Systems*, pages 127–138. San Diego, 1992.
14. D. Jana and Bharat Jayaraman. Set constructors, finite sets, and logical semantics. *Journal of Logic Programming*, 38(1):55–77, 1999.
15. Bharat Jayaraman and K. Moon. Subset logic programs and their implementation. *Journal of Logic Programming*, To appear:?–?, 1999.
16. Jürgen Dix Jose Arrazola and Mauricio Osorio. Confluent rewriting systems in non-monotonic reasoning. *Computacion y Sistemas*, Volume II, No. 2-3:104–123, 1999.
17. R. Kowalski. *Logic for problem solving*. North Holland Publishing Company, 1979.
18. R. Kowalski and F. Toni. Abstract Argumentation. *Artificial Intelligence and Law Journal*, pages 275–296, September 1996.
19. G. M. Kuper. Logic programming with sets. *JCSS*, 41(1):44–64, 1990.
20. M. Liu. Relationlog: A Typed Extension to Datalog with Sets and Tuples. In John Lloyd and Evan Tick, editors, *Proceedings of the 1995 Int. Symposium on Logic Programming*, pages 83–97. MIT, June 1995.
21. John W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, 1987. 2nd edition.
22. Bharat Jayaraman Mauricio Osorio and J. C. Nieves. Declarative pruning in a functional query language. In Danny De Schreye, editor, *Proceedings of the International Conference on Logic Programming*, pages 588–602. MIT Press, 1999.
23. Bharat Jayaraman Mauricio Osorio and David Plaisted. Theory of partial-order programming. *Science of Computer Programming*, 34(3):207–238, 1999.
24. E. Mendelson. *Introduction Mathematical logic (3th ed.)*. Wasdworth and Brooks/Cole Advanced Books and Software, United Stated, 1987.
25. Mauricio Osorio. Semantics of partial-order programs. In J. Dix and L.F. del Cerro andU. Furbach, editors, *Logics in Artificial Intelligence (JELIA '98)*, LNCS 1489, pages 47–61. Springer, 1998.
26. Mauricio Osorio and Bharat Jayaraman. Aggregation and negation-as-failure. *New generation computing*, 17(3):255–284, 1999.

27. Mauricio Osorio and J. C. Nieves. Extended partial-order logic programming. In Gianfranco Rossi and Bharat Jayaraman, editors, *Proceedings of the Workshop on Declarative Programming with Sets*, pages 19–26, Paris, France, 1999.
28. Mauricio Osorio and Erika Saucedo. Aplicación de wfsx en el campo de la botánica. In *Proceedings of II Encuentro Nacional de Computación*, pages ?–? Mexico, 1999.
29. Mauricio Osorio and Fernando Zacarias. High-level logic programming. LNCS 1762, pages ?–?, Berlin, 2000. Springer Verlag.
30. K. Ross. Modular stratification and magic sets for datalog programs with negation. *Journal of the ACM*, 41(6):1216–1266, 1994.
31. K.A. Ross and Y. Sagiv. Monotonic aggregation in deductive databases. In *Proc. 11th ACM Symp. on Principles of Database Systems*, pages 114–126. San Diego, 1992.
32. Allen Van Gelder. The Well-Founded Semantics of aggregation. In *Proc. of the 11th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Diego, USA*, pages 127–138. ACM Press, 1992.

Experiments on Information Retrieval Using Case-Based Reasoning

Carlos Ramirez

ITESM Campus Querétaro
Epigmenio Gonzalez 500, San Pablo, Querétaro, Qro. México
Ph. +52 4 2383280
Fax: +52 4 2383279
{cramirez@campus.qro.itesm.mx}

Abstract. This paper presents a demonstration of a Theory for Episodic Learning applied to Information Retrieval. The theory is compatible with Case-Based Reasoning and is used to model how contextual information about users, their aims and search-sessions experience may be represented in case bases, and automatically tuned up. The proposed model shows how query processing may be extended by deriving information from the case bases, taking advantage of user profiles and a history of queries derived from search-sessions. The paper includes a demonstration of the program IRBOC, presenting a fully worked example.

1 Introduction

Document representation and retrieval strategies have been the centre of attention in information retrieval research for many years. There has been concern in the representation of the users' information needs, as well as in the way users interact with information retrieval systems (Belkin et al., 1993a, 1993b, 1993c; Ingwersen, 1986; Croft, 1987; Croft and Thompson, 1987); but there is still plenty of room for investigation of these very important issues, the user representation needs and their interpretation. In this paper, it is suggested that a comprehensive and integrated view of the problem is needed. A novel architecture is proposed based on a Theory of the Acquisition of Episodic Memory (Ramirez and Cooley, 1997), compatible with Case-Based Reasoning (CBR), particularly concerned with the representation and interpretation of the users' information needs, which makes use of information about the context of information requests. The rationale for this architecture is not new, Belkin (1977, 1980), De Mey (1980), and Watters (1989), amongst others have been concerned with the need to support semantic and contextual information in the retrieval process; but CBR had not been used before for this purpose. It is maintained here, that CBR is a promising technology for tackling problems in information retrieval.

2 The Retrieval of Information and the User

The goals, projects, research interests, and other background information about users are relevant pieces of information that should not be disregarded. They can be

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 25—39, 2000.

combined with query information and used in a framework in which documents can be retrieved with increased accuracy. Hence, an important goal of this work is to show that the users' background knowledge and the context of search-sessions are central to the effectiveness of an information retrieval system, and that CBR has the potential to handle them.

2.1 The User's Integrated Information Request

In order to represent effectively an information request, all available information, in addition to the query, that may affect the retrieval process should be taken into account. Some of this information can be provided by the user (in addition of the query) at the time of the request, but not all of it, since it would be too unpractical—providing it was found a way to represent and deal with the information. However, most of this context information, that originates in the background of the user, does not usually change in the short term, so it is feasible to enter it into the system beforehand, store it in individual profiles, and use it when required. Also, additional context information can be obtained from records of interactions. In this paper, it is emphasised the use of context in addition of the query, in the formation of the *user's integrated information request*.

IRBOC, an information retrieval system, was designed having in mind specialised libraries and document centres. Users of IRBOC are expected to be working in specific projects, or at least have clear lines of work or research. This is not a restriction imposed by the model, but a constrain for effectiveness. In order to use past search-session cases, and particularly, in order to define and organise a useful context, the topics of this context have to be constrained. IRBOC performs better and improves quicker when working with context related to specific domains. Although, there is no restriction in the number of user-profiles that a user can have, which means that a specific user can work on different domains and simply have different *user_ids* for each domain.

2.1.1 The Query

Query (Q) is the users' explicit description of their immediate information needs. A query can be expressed in several ways, for example, as a Boolean expression, as a natural language sentence, or as a simple list of keywords. The specification of the query is intended to express the content and type of the information needed. Each form of representation has its advantages and disadvantages, ample research has been carried out comparing different representations (Salton and McGill, 1983; Forsyth and Rada, 1986); but it is impossible to generalise such results for every retrieval system, since conditions are simply different on each situation.

Nevertheless, the specification and representation of queries has been a controversial topic in Information Retrieval (IR), due to the fact that formulating requests is an ambiguous task. It is ambiguous because of the following factors, amongst other of a less intangible nature:

- Users perception and description of their information need may change in the course of their interaction with the system (i.e., Belkin's [1980, 1993] "anomalous states of knowledge").

- Users may not be certain of their information needs; they are typically looking for an idea, rather than for specific documents.
- Users typically want 'to see what has been said about something' rather than request 'what X has to do with Y' which is significantly more precise (see Sparck Jones, 1990).

2.1.2 Context

The *context* (C) of a request is all the information that plays a roll in the retrieval process (i.e., it is related to the target documents), but is not expressed in the query. Context allows a better and more complete understanding of the information need. Context is the information that a user usually has in mind during the interaction with the retrieval system, but that is not allowed to express in the query, because conventional systems do not have the capabilities to do it; such as research interests, information intentions, goals, etc. In general, the user's background can not be stated in the query, although it may play an important roll in the retrieval process. Those non-explicit elements of the information need are relevant pieces of information that should not be disregarded because they allow a better understanding of the users' information needs. The context combined with the query give a more complete framework, in which searches of documents can be carried out with increased accuracy. It is the thesis of this work, that CBR has the potential to handle them, to improve the effectiveness of IR systems.

If a search-session (i.e., a user interacting with an IR system) could be viewed as a communication event, then context is the background[1] of the receptor of a communicate. A background shared with the receptor (constituted with similar knowledge), such that the receptor is capable to recognise and process the intentions of the transmitter. Without any common background, the receptor could be unable to agree about the intentions of the communicate. For instance, without a context, the sentence: "How can I get free ranch eggs?" may be understood quite differently by a farmer, by a foreigner, and by the owner of a grocery store. The reason is that everyone tries to understand the phrase according to his or her background.

Human intermediaries between an IR system and its users, routinely use context information to carry out 'searches', but they do not explicitly incorporate the context into the system, they use it implicitly to make inferences that later result in the formulation of a specific query. For example, if the user happens to be a child interested in literature, then the intermediary will assume that the user is interested in children's literature, and not in James Joyce or Borges' novels.

3 Architecture of the Model

The conceptual model is based in Ramirez and Cooley's (1997) theory of the acquisition of episodic memory. The Information Retrieval System Based On Cases (IRBOC) is organised around a case base of user-profiles, a case base of search-

[1]Maturana (1978) calls it a *concensual domain*. It is assumed that the domain is co-operatively constituted by all the active members of a particular domain.

sessions, a CBR module, an IR module, and a database of documents. The *IR Module* is based on standard Boolean logic (AND, OR, and NOT operators), plus the truncation or wild card (*) and adjacent (ADJ) operators, parenthesis are also allowed. The CBR module functions are: the creation and maintenance of *user-profile cases* (UPC) and *generalised user-profiles* (GUPC), based on information stored in a group of containers called *user's context components;* the creation and maintenance of the *search-session cases* (SSC) and the *generalised search-session cases* (GSSC), which are automatically developed during the normal use of the system; and finally, CBR methods are used to exploit the case bases, in order to modify the initial query and also suggest alternative queries. The database of documents (i.e., the set of target documents) contains surrogate documents on a specific domain of knowledge. The architecture is shown in Figure 1.

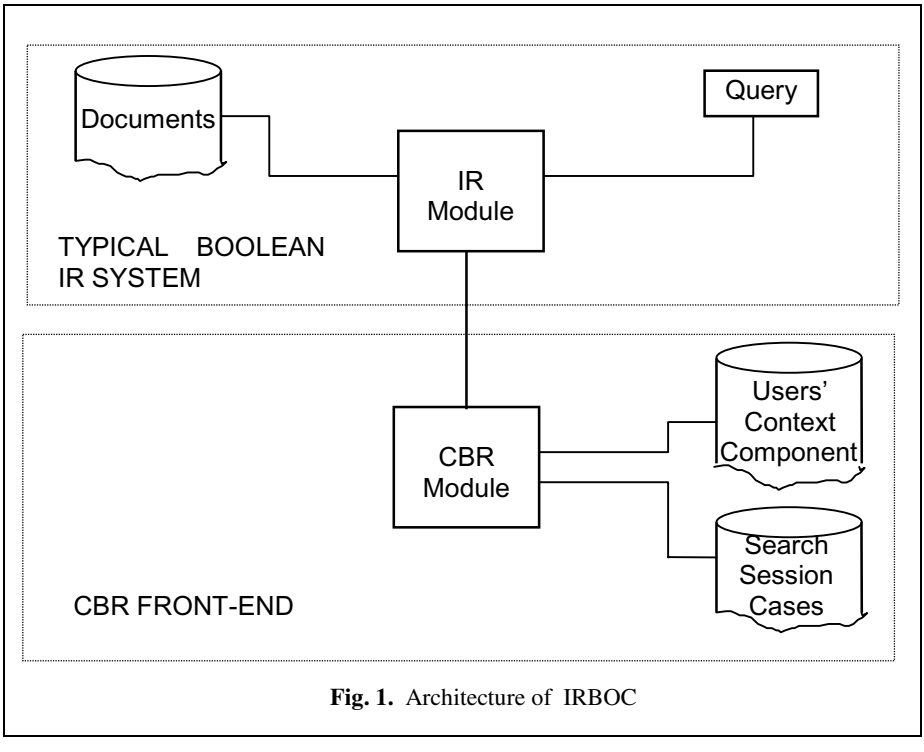


Fig. 1. Architecture of IRBOC

4 Design Issues

4.1 Context Organisation

Context information has been organised in the following *context components*: User's Background, Documents' Profile, Research Projects, Research Interests, and a Search-Sessions component. They represent the most relevant information concerning the information needs of the users related to specific domains, besides their queries. Each component is organised by attributes, which contain groups of terms. Not all the

attributes have the same relevance, so there is an independent weight associated with each of them (a real number between 0 and 1) defined in the general parameters of the system. Also, each term has an individual score associated to each term (an integer number between 1 and 5) assigned by the user. The components are organised as follows:

4.1.1 User's Background Component

This component is formed with the profile of every user. A profile is formed with the following attributes:

- Occupation or activities of the user
- Current areas of work
- Related areas
- Areas of specialisation
- Other areas of experience

The attributes of this component are used to complement the other context components. Notice that there is no need to update this component with every request, but only when the user's main activities change.

4.1.2 Projects

This context component is used to represent the aims of the information requested, and should respond to the question: What is the information going to be used for? The following attributes form this component:

- Current project
- Project objectives
- Goals related to the project
- Past projects

Notice that as the previous component, there is no need to update this component with every request.

4.1.3 Research Interests

This component contains information about the personal goals of the users, concerning the current request:

- Topics of interest related to the current query
- Other topics of interest related to the current query

This is the most temporal of the components, and should be updated with every request. The attributes of this component have the highest weight when used to modify or expand the query.

4.1.4 Documents' Profiles Component

This is an optional component, whose information is helpful to evaluate the relevance of retrieved candidate documents by filtering those documents that do not satisfy the constrains of style, type, and level of technical complexity requested by the user; however, the creation of this component involves additional cataloguing work. A document profile contains information about the style or approach of a document,

e.g., descriptive, narrative, academic, scientific, or informal, about the type of document, e.g., theoretical, experimental, field notes, etc., and about the level of technical complexity of the document. These attributes are useful to filter documents during the process of been retrieved. For example, an experienced user knows that papers from Proceedings of Scientific Conferences may suit certain needs of researchers, but it is less likely to suit the needs of technicians, or the needs of an undergraduate student preparing a subject-examination. A list of attributes for this component follows:

- The approach or style (e.g., academic, scientific, descriptive, narration, informal, etc.).
- The type of document (e.g., theoretical, experimental, field notes, technical report, bibliographic research, etc.).
- Level of technical complexity (e.g., low, medium, high).

4.1.5 Query History Component

As part of the context components, this component is used to keep track of all successful search-sessions. Records are created during the normal use of the system and stored for future usage, including pointers to specific documents, to specific users, and to a particular profile (generalisation of users). Each search-session is formed with the following attributes:

- Initial query
- Intermediate queries
- Final query
- Results (indices to documents)

4.2 The Organisation of Case Bases

4.2.1 User-Profiles Case Base

The user-profile case base is created with attributes from three of the context components: Background, Research Interests, and Projects. Similarities between the attributes of the components of different users are the basis for the formation of UPCs. A special similarity function for IRBOC was devised after Tverky's "Contrast Model of Similarity" (Tversky, 1977). The function was modified in such a way that contextual information influences—but not controls—the similarity metric (see also Cain, et al., 1991), and the necessary adjustments to fit IRBOC architecture. The metric function is shown below.

4.2.2 Generalised User-Profiles Case Base

The generalised user-profile case base is created with clusters of user-profiles cases, as presented in Figure 2. Similarity between the attributes of clusters of user-profiles cases is the basis for this process; similarity is computed using (2), shown in section 4.5.

4.2.3 Search-Sessions Case Base

A case base of search-session cases is created with the successful queries carried out during search-sessions, plus a link to the corresponding user-profile, which allows access to the context component associated to the posed queries.

4.2.4 Generalised Search-Sessions Case Base

Two case bases of generalised search-session cases are created with common attributes obtained from clusters of similar search-session cases. One of the case bases is based on the Projects component, and the other case base is based on user-profile user-profiles, as shown in Figure 2. The GSSCs are used to identify useful queries based only on the topic, and based on the general context of the request, respectively.

4.3 The Information Retrieval Module

Queries are posed to the IR system using keywords and standard Boolean logic (NOT, AND, OR). The operators adjacent (ADJ) and like or truncation (*) are also implemented. Parenthesis can be used to alter the default precedence of the Boolean operators, as usual. A conventional default stemming process (Frakes and Baeza-Yates, 1992) is applied to each of the keywords not displaying the truncation operator. Also, a standard list of stop-words is used to discard irrelevant words from the query. The system has various programs that allow the user to easily enter and update the context components when required. The main program has an interface that guides the user all the way through the search-session; details of the user-profile can be displayed; previous queries can also be displayed. During the search-session, queries are sequentially displayed and the corresponding expanded queries can also be seen if required. Once documents are retrieved and inspected, the relevant ones can be marked to indicate that the query and terms were successful, so IRBOC can take actions reinforcing the relevance of the corresponding terms.

4.5 The Case-Based Reasoning Module

The goal of this research is to enhance a conventional Boolean retrieval system by using a novel architecture based on CBR. In IRBOC, CBR methods are used in two ways: firstly, by using contextual information about the user to modify and expand the query; and secondly, by taking advantage of previous search-sessions. CBR methods are used to create and maintain user-profile cases and generalised user-profile cases, which are used to expand the query with new keywords. Search-session cases and generalised search-session cases are used to suggest alternative queries to specific requests.

UPCs are formed by associating the terms related to a specific user-id from the various context components. Term weights are the product of multiplying the term scores by the attribute scores (score values are integers between 1 and 5) and then

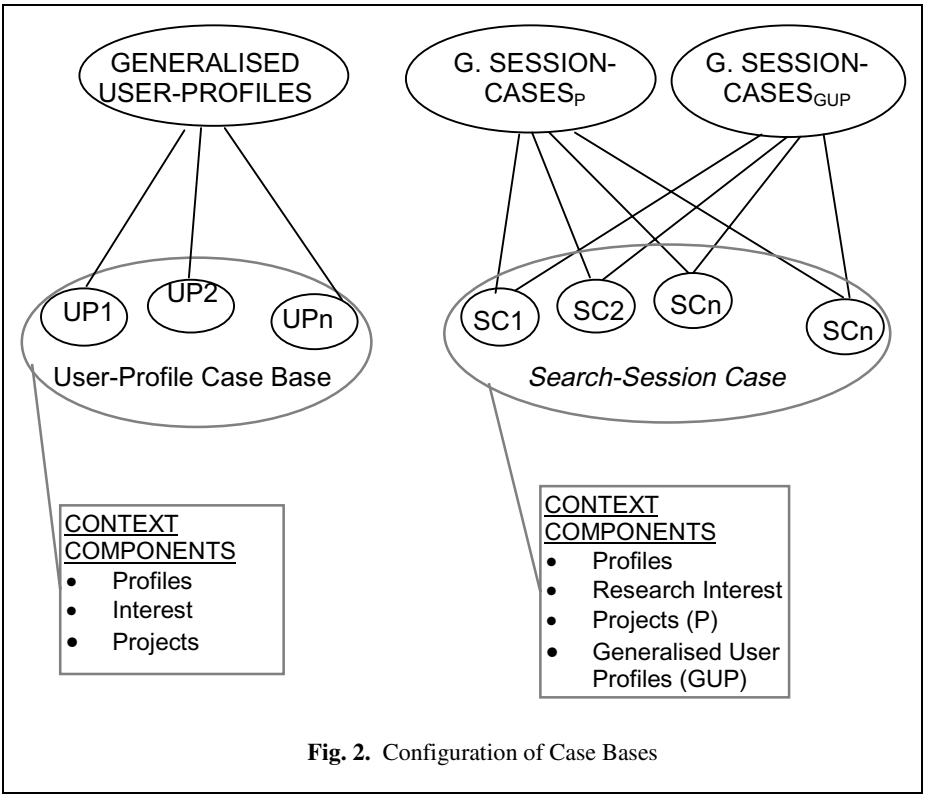


Fig. 2. Configuration of Case Bases

dividing by 25 to normalise the result (to a real number between 0 and 1), as shown in (2.2). GUPCs are created or modified depending on their similarity distance to new UPCs. The distance of all existing GUPCs against a new UPC is computed using (2) and then compared against a threshold. The GUPC with the lowest value (*Dist* values are real numbers between 0 and 1) is chosen, provided the threshold is greater than the value. The similarity distance metric function (2) was specially designed for IRBOC (as explained above), where *A* represents the a UPC and *B* represents a GUPC, *A* and *B* are attribute-sets of cases *a* and *b*. Parameters θ, α, and β represent the relative importance of the common and distinctive attribute-sets, so for instance, more weight can be given to common attributes. The values of these parameters can be easily reset depending on the performance of the system. The effects of these parameters also make a positive effect in the asymmetries that appear in similarity judgements. Tversky indicates that in similarity statements there is a subject and a referent, we say that "*a* [subject] is like *b* [referent]", where the most prominent object usually been the referent (see Tversky, 1977, and also Tversky and Gati, 1978). In general IRBOC's formula predicts that as the number of common attributes increases and the number of distinctive attributes decreases, the two objects *a* and *b* become more similar.

The first part of the formula computes the similarity between common terms (2.1). If there are no common terms at all (k=0), then $dis(a,b) = 1$, which is the

maximum possible distance between a and b, otherwise only those terms that are common to *A* and *B* are considered, since *f(a,b)* becomes 0 when $a \neq$ b. The second part and third parts of the formula compute the distinctiveness of *a* and *b* respectively, by considering only non-common terms (2.3).

A particularity of this metric is that is not sensitive to the dimensions of the attribute-sets *A* and *B*, in opposition to other widely used functions, such as the similarity function in the CBL1-4 algorithms, which assume maximum difference from the value present when attribute values are missed (Aha et al., 1991, Aha, 1991). Or nearest neighbour functions such as Remind's (Cognitive Systems 1992, presented in Kolodner, 1993), which does not even make provision for dealing with this situation. For a comparative study of similarity functions see Ramirez (1998), Wettschereck and Aha (1995) and Willet (1988).

Other function of the CBR processor is the maintenance of the case bases (i.e., modification and deletion of cases, and the optimum re-organisation of their structures); this task involves analysing the effectiveness of cases, which is carried out based on their success and frequency of use. A usage counter is used for this purpose, which is attached to each term.

$$Dist(A,B) = \theta \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} dis(a_i,b_j)}{k} + \alpha \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} diff(a_i,b_j)}{n \times m} + \beta \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} diff(b_j,a_i)}{m \times n} \quad \textbf{(2)}$$

where

$$dis(a,b) = \begin{cases} d(a,b) & k > 0 \\ 1 & k = 0 \end{cases} \quad \textbf{(2.1)}$$

$$d(a,b) = |w(a) - w(b)| \cdot f(a,b)$$

$$f(a,b) = \begin{cases} 1 & \text{a = b} \\ 0 & \text{otherwise} \end{cases}$$

$$w(a) = 2\frac{W_{Term}(a)}{10} \cdot 2\frac{W_{Attribute}(a)}{10} \; ; \quad w(b) = 2\frac{W_{Term}(b)}{10} \cdot 2\frac{W_{Attribute}(b)}{10} \quad \textbf{(2.2)}$$

$$diff(a,b) = w(a) \times g(a,b); \quad g(a,b) = \begin{cases} 1 & a \neq b \\ 0 & a = b \end{cases} \quad \textbf{(2.3)}$$

where $\theta, \alpha, \beta \in$ R, and $0 < \theta \leq 1, 0 < \alpha \leq 1, 0 < \beta \leq 1,$ and $\theta + \alpha + \beta = 1.$

θ represents the relative importance of common terms.

α represents the relative importance of distinctive terms in B.

β represents the relative importance of distinctive terms in A.

$a \in A$, $b \in B$.

$k =$ Number of common terms, where $a_{ij} = b_{ij}$; $k = 1$ iff $k = 0$.

$n =$ Number of terms in set *A*.

$m =$ Number of terms in set *B*.

Queries are expanded by selecting the best terms from the corresponding user-profile case, and from the corresponding generalised user-profile case, and even from other cases related to that specific GUPC, if necessary. And finally, IRBOC is also capable of proposing alternative new queries. Those new queries are obtained from previous similar ones stored in the search-session case base, by comparing the UPC and the Projects component with the GUPCs.

5 IRBOC in Action: An Example

A postgraduate student with id 'PG-015' is looking for research documents about work carried out in AI applied to information retrieval. The user needs the information to find out what has been done in that area of research, and is particularly interested in working systems. The user has used the system before and the context components have been already defined as shown below. The pairs following each term are the weight of the term and a counter of usage (the number of times that the term has been successfully used):

User-Profile Component:
- User Id: PG-015
- Occupation or activities: postgraduate(3,0) student(1,0); research(4,0)
- Current areas of work: 'computer science'(2,0); 'artificial intelligence'(4,0); 'information science'(2,0); 'cognitive science'(2,0)
- Related areas: 'machine learning'(4,0); 'expert systems'(4,0); databases(2,0); 'cognitive psychology'(2,0)
- Areas of specialisation: 'case-based reasoning'(5,0); learning(2,0); 'information retrieval'(5,0); databases(3,0)
- Other areas of experience: librarianship(3,0); linguistics(3,0)

Projects Component:
- User Id: PG-015
- Current project: case-based reasoning(5,0); information retrieval(5,0)
- Project objectives: exploration(3,0); intelligent(5,0); information retrieval systems(5,0); improve(4,0); enhance(4,0); effectiveness(3,0)
- Goals related to the project: case-based-reasoning research(4,0); information-retrieval exploration(3,0)
- Past projects: conceptual(2,0); model(4,0); learning(4,0); episodic memory(5,0) memory structures(5,0); CBR foundations(5,0)

Research Interests component:
- User Id: PG-015
- Topics of interest related to the request: case-based reasoning(5,0); information retrieval(5,0); expert systems(4,0); automatic document processing(3,0); artificial intelligence(4,0); context models(4,0)
- Other topic of interest: natural language processing(2,0); language ontology(4,0); cognitive science(5,0); cognitive psychology(4,0); memory(2,0); perception(2,0); learning(4,0); understanding(3,0); explanation(3,0); reasoning(3,0); analogy(4,0); similarity measure(4,0); concept theories(3,0); categorisation(3,0); knowledge acquisition(4,0); knowledge

representation(4,0); dynamic memory theory(5,0); automatic text processing(4,0); automatic document indexing(3,0); document transformations(3,0); matching(4,0); term weighting(4,0); relevance feedback(3,0); search strategies(4,0); clustering(3,0); automatic classification(4,0)

Document Profile:
The approach or style: scientific
The type of document: any
Level of technical complexity: medium-high

Query:
(Automatic OR Intelligent) AND information ADJ retrieval AND systems

Procedure:

1. Using the information of the context components and the query, the CBR processor matches two similar queries stored in the search-session case base. Those two queries were posed to IRBOC by another user whose Project component contained several terms in common with the current user's query. The obtained queries, which are suggested to the user are:

1. "Automatic ADJ Document ADJ Processing"
2. "Expert ADJ Systems AND Information ADJ Retrieval"

2. The user decides not to pursue the suggested queries, then the CBR processor tries to re-formulate the initial query as follows:
a) First, the terms with the best scores are retrieved from the corresponding UPC, providing the normalised scores satisfy a threshold.
b) The query is expanded with the retrieved terms [1] and documents are retrieved.
c) If the user is satisfied with the documents the search session ends here, otherwise the search-session continues, and the associated GUPC to the current UPC is used to retrieve all of the other associated UPCs.
d) The best UPC is selected.
e) The associated terms with the best scores are used to expand the initial query [2] and new documents are retrieved.
f) The user marks those documents that consider relevant and the search-session finishes here, although it could continue if the user wants to modify the initial query.

Initial Query[2]: (automatic | intelligent)^information+retrieval+systems
Expanded Queries:
[1] (automatic | intelligent) ^ information+retrieval+systems | (artificial+intelligence | expert+systems | machine+learning |)
[2] (automatic | intelligent) ^ information+retrieval+systems | (case-based+reasoning | explore | enhance | improve | document-processing)

[2]Where +stands for ADJACENT, ^ stands for AND, | stands for OR, and ¬ stands for the NOT operator.

Documents retrieved in [1]: B30650, B89899, B5897, B32001, B56564, B25668.
[2]: B14683, B6784, B25668, B15688.

The titles of the documents (books) retrieved and marked as relevant are:

B30650: “Expert Systems for Reference and Information Retrieval”

B5897: “Automatic Text Processing”

B32001: “Commercial Information Retrieval Systems: An information managers guide”

B14683: “The SMART retrieval system: Experiments in automatic document processing”

B6784: “Information Retrieval Experiment”

B25668: “Machine Learning: Applications in expert systems and information retrieval”

3. The above two queries constitute a new search-session case and are added to the Query History component, links to documents and SSC indices are also updated.

4. Since the example presented here actually corresponds to a new user, the system creates a new user-profile case, and a new search-session case.

5. The UPC is added to the corresponding Generalised-UPC . That is, new terms from the UPC are added to the GUPC.

6. The corresponding generalised search-session case is modified with the term “case-based reasoning” only, originated from the Projects component.

7. Finally, the weights and usage-counter of the terms used to modify the query are updated (reinforced).

Summarising the changes to memory: IRBOC learned a new user-profile, a new search-session case, and tuned up two generalisations.

6 Evaluating IRBOC’s Retrieval Module

Evaluation measures used in IRBOC are based on the well-known concepts in Information Retrieval of recall and precision (Salton and McGill, 1983). A set of experiments was carried out using a data set of 20 User Profile Cases. The first set of tests was carried out using the Boolean Retrieval System alone, a second set was carried out using the "Query Expansion Method" of the CBR front-end, and a third set of tests using the "Substitution Method". The tests conditions used in all the experiments reported here were the same, including the CBR system parameters. The configuration of the CBR front-end was as follows: there were 20 user-profile cases stored, 11 generalisations of user-profiles, and 26 search-session cases. Runs were carried out for users “AC-001,” “AC-003,” “AC-004,” “AC-005,” “AC-009,” and “AC-014,” because those were the only users associated with proper generalisations. The results are shown in Table 1, including the results of the 3 sets of tests, which are shown separated by ‘/’. For example, Query No. 1 retrieved 6 documents using the Boolean retrieval system without the CBR front-end, and 19 documents after applying the “query expansion” method of the CBR front-end, and 21 using the "query substitution" with expansion. The results clearly indicate an important increase in the

recall variable. From the 25 queries, 21 retrieved more relevant documents than using stand-alone queries, 1 retrieved less documents, and 3 the same number. From the 21 queries that increased their recall, 4 had not recalled any documents in their stand-alone form. The recall average indicates an important increase in most queries.

Query	User Id	GUPC Cluster Id	No. of Docums. Retrieved	No. of Relev. Docum. In the Collection	No. of Relevant Docums. Retrieved	Recall	Precision
1	AC-001	1	6/19/21	12	5/11/11	.42/.92/.92	.83/.58/.52
2	AC-001	1	0/20/21	12	0/6/5	0/.50/.42	0/.30/.24
3	AC-001	1	4/21/24	12	3/6/7	.25/.50/.58	.75/.29/.29
4	AC-003	1	5/19/21	8	4/5/4	.50/.62/.50	.80/.26/.19
5	AC-003	1	2/21/24	8	2/2/2	.25/.25/.25	1.0/.10/.08
6	AC-003	1	5/19/21	8	4/5/4	.50/.62/.50	.80/.26/.19
7	AC-003	1	9/20/20	8	6/6/6	.75/.75/.75	.67/.30/.30
8	AC-004	3	11/30/32	93	8/12/17	.09/.13/.18	.73/.40/.53
9	AC-004	3	4/28/32	93	2/9/11	.02/.10/.12	.50/.32/.34
10	AC-005	3	61/30/32	112	14/12/9	.12/.11/.08	.23/.40/.28
11	AC-005	3	4/30/32	112	3/5/7	.03/.04/.06	.75/.17/.22
12	AC-005	3	1/30/32	112	1/1/3	.01/.01/.03	1.0/.03/.09
13	AC-005	3	2/34/36	112	1/4/5	.01/.04/.05	.50/.12/.14
19	AC-009	7	4/30/33	29	3/6/7	.10/.21/.24	.75/.20/.21
20	AC-009	7	9/30/33	29	8/9/9	.28/.31/.31	.89/.30/.27
21	AC-009	7	0/28/30	29	0/9/7	0/.31/.24	0/.32/.23
22	AC-009	7	11/30/33	29	7/15/17	.24/.52/.59	.64/.50/.52
23	AC-014	7	11/33/34	29	3/7/8	.10/.24/.28	.27/.21/.24
24	AC-014	7	4/31/32	29	2/5/6	.07/.17/.21	.50/.16/.19
25	AC-014	7	1/33/34	29	0/6/6	0/.21/.21	0/.18/.18
26	AC-014	7	0/28/28	29	0/3/3	0/.10/.10	0/.11/.11

Table 1. Comparing the results of three methods of retrieval in IRBOC: stand-alone Boolean queries, "Query Expansion," and "Query Substitution" with expansion.

7 Conclusions

A computational implementation of Ramirez's Theory for Episodic Learning proved to be feasible. This is an important advancement for the foundations of CBR, since previous theories, such as Schank's Theory of Dynamic Memory and particularly other theories for learning proposed in Psycology, such as Medin and Smith (1984) and Pazzani and Silverstein (1990) do not have enough detail in their conceptualisation as to demonstrate them in computer programs.

On the other hand, the experiments reported in this paper shown that CBR is usefull to tackle problems in Information Retrieval, an area were information goals are frequently underspecified or even unclear for the user.

IRBOC's results were very representative, specially when compared to a plain Boolean Information Retrieval. It is expected that the effectiveness of the system will improve further in time, with the acquisition of additional, accurate experience through new search-sessions.

To conclude this paper, is worth mentioning that although further experimentation is required to tune up the system (by finding out the optimum values of weights for the context components, the best values for the tuning parameters of the similarity and generalisation functions, and determining the most effective thresholds), current results are promising.

References

Aha, D. (1991). Case-Based Learning Algorithms. In R. Bareiss (Ed.), *Proceedings: Case-Based Reasoning Workshop* / DARPA, Washington, D.C., Morgan Kaufmann, 147-158.

Aha, D., Kibler, D., and Albert, M.K. (1991). Instance-Based Learning Algorithms. *Machine Learning*. 6, 37-66.

Belkin, N.J. (1980). Anomalous States of Knowledge as a Basis fir Information Retrieval. *The Canadian Journal for Information Science*, 5, 133-143.

Belkin, N.J. and Cool, C. (1993a). The Concept of Information Seeking Strategies and its Use in the Design of Information Retrieval Systems. *Case-Based Reasoning and Information Retrieval: Exploring the opportunities for technology sharing*. Papers from the 1993 Spring Symposium, Technical Report SS-93-07, AAAI Press, 1-7.

Belkin, N.J., Cool, C., Stein, A., and Thiel, U. (1993b). Scripts for Information Seeking Strategies. *Case-Based Reasoning and Information Retrieval: Exploring the opportunities for technology sharing*. Papers from the 1993 Spring Symposium, Technical Report SS-93-07, AAAI Press, 8-17.

Belkin, N.J., Marchetti, P.G. and Cool C. (1993c). BRAQUE: Design of an interface to support user interaction in information retrieval. *Information Processing and Management*, 29(4): 325-344.

Cain, T., Pazzani, M.J., and Silverstein, G. (1991). Using Domain Knowledge to Influence Similarity Judgements. In R. Bareiss (Ed.), *Proceedings: Case-Based Reasoning Workshop* / DARPA, Washington, D.C., Morgan Kaufmann, 191-199.

Croft, W.B. (1987). Approaches to Intelligent Information Retrieval. *Information Processing and Management*, 23(4): 249-254.

Croft, W.B. and Thompson, R.H. (1987). I3R: A new approach to the design of document retrieval systems. *Journal of the American Society for Information Science*, 38(6), 389-404.

De Mey, M. (1980). The relevance of the cognitive paradigm for information science. In O. Harbo and L. Kajberg (Eds.), *Theory and Application of Information Research, Proceedings of the Second International Research Forum on Information Science*, Mansell, London.

Frakes, W.B., and Baeza-Yates, R. (Eds.) (1992). *Information Retrieval: Data Structures and Algorithms*. Englewood Cliffs, NJ, USA: Prentice Hall.

Forsyth, R. and Rada, R. (1986). *Machine Learning: applications in expert systems and information retrieval*. Chichester, UK: Ellis Horwood.

Ingwersen, P. and Pejtersen, A.M. (1986). User requirements—empirical research and information systems design. In Peter Inwersen (Ed.), *Information Technology and Information Use*. London: Taylor Graham, 111-125.

Maturana, H.R. (1978) Biology of Language: The Epistemology of Reality. In George Miller and Elizabeth Lenneberg (Eds.), *Psychology and Biology of Language and Thought: Essays in honour of Eric Lenneberg*. Academic Press, 27-63.

Medin, D.L. and Smith, E.E. (1984). Concepts and concept formation. *Annual Review of Psychology*, 35: 113-138.

Pazzani, M. and Silverstein, G. (1990). Learning from examples: The effect of different conceptual roles. *In Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*. Cambridge, MA: Earlbaum, 221-228.

Ramirez, C. (1998). A Theory of Episodic Memory for Case-Based Reasoning and its Implementation, *Ph.D. Thesis, University of Kent at Canterbury*, Kent, UK.

Ramirez, C. and Cooley, R. (1997). A Theory of the Acquisition of Episodic Memory. In D. Aha and D. Wettschereck (Eds.), *9th European Conference on Machine Learning: Workshop Notes on Case-Based Learning: Beyond Classification of Feature Vectors*, Prague, Czech Republic, 48-55.

Salton, G. and McGill, M.J. (1983). *Introduction to Modern Information Retrieval*. New York: McGraw Hill.

Sparck Jones, K. (1990). Retrieving information or answering questions?, London, British Library (The Eighth British Library Annual Research Lecture 1989).

Tversky, A. (1977). Features of similarity. *Psychological Review*, 84, 327-352.

Watters, C.R. (1989). Logic Framework for Information Retrieval. *Journal of the American Society for Information Science*, 40(5), 311-324.

Wettschereck, D. and Aha, D.W. (1995). Weighting Features. In M. Veloso and A. Aamodt (Eds.), *Case-Based Reasoning Research and Development; Proceedings / ICCBR*-95, Portugal, Springer-Verlag, 347-358.

Willet, P. (1988). Recent Trends in Hierarchic Document Clustering: A Critical Review. *Information Processing and Management*, 24(5): 577-597.

A Probabilistic Exemplar-Based Model for Case-Based Reasoning

Andrés F. Rodríguez[1], Sunil Vadera[2], and L. Enrique Sucar[3]

[1] Instituto de Investigaciones Eléctricas
AP 475, Cuernavaca, Mor. México
afrm@iie.org.mx
[2] School of Sciences, University of Salford
Salford, M5 4WT, UK
S.Vadera@cms.salford.ac.uk
[3] Departamento de Computación, ITESM - Campus Morelos
AP C-99, Cuernavaca, Mor., 62020, México
esucar@campus.mor.itesm.mx

Abstract. An exemplar-based model with foundations in Bayesian networks is described. The proposed model utilises two Bayesian networks: one for indexing of categories, and another for identifying exemplars within categories. Learning is incrementally conducted each time a new case is classified. The representation structure dynamically changes each time a new case is classified and a prototypicality function is used as a basis for selecting suitable exemplars. The results of evaluating the model on three datasets are presented.

1 Introduction

Organizing and indexing cases in memory is a fundamental part of *case-based reasoning* (CBR) that involves learning and reasoning processes. This problem can be divided into two parts. The first is the selection of the features of the cases that can be used to index and retrieve the cases. The second is the organisation of the case memory so that the case retrieval process is efficient and accurate.

One approach that has been used to address this problem has been to store all the cases and develop algorithms to partition the search space for retrieving similar cases. So for example, systems like REMIND provide a tree induction algorithm that can be used to avoid examining all the cases. This kind of approach is particularly useful when large databases of cases are already available. However, when cases are not available in advance, and the domain is not well defined this approach is more difficult to apply.

An alternative approach, that is perhaps more applicable to such situations, is to store only prototypical cases. This approach, known as the *exemplar-based* model has its basis in cognitive theories, which postulate that concepts can be represented by exemplars [13,14]. However, previous implementations of the exemplar-based models have struggled to produce systems that can be justified in a rational manner. For example, the Protos system [3] uses many heuristics

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 40–51, 2000.

and mechanisms for combining evidence that are hard to justify independently of its original application.

Hence, this paper attempts to develop an exemplar-based model with foundations that utilise Bayesian models.

2 Exemplar-Based Models: The Problem

An exemplar-based model is thought to be particularly appropriate for weak domains [11], where it is difficult to define categories by explicitly using classical membership constraints. For such situations, which can be common in real applications, an exemplar-based model may provide a better representation of the categories. Figure 1 illustrates the idea of a weak domain together with cases, exemplars, and categories. It shows two categories, A, B (the solid lines), that each have exemplars that represent regions (the dashed lines) that contain cases (the dots).

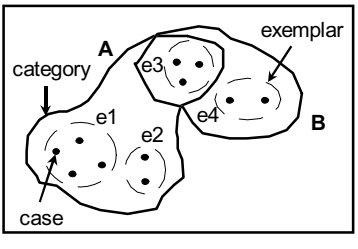


Fig. 1. Cases, exemplars and categories in a weak domain.

To illustrate the problem, suppose that the category A is represented by the exemplars e_1, e_2, and e_3 and the category B is represented by the exemplars e_3 and e_4. Also suppose that the exemplars e_1, e_2, e_3, and e_4 currently represent 4, 2, 3, and 2 cases respectively. Now suppose that a new case is given. The following two functions must be provided by an exemplar-based model:

1. Determine the exemplar that best classifies the new case given the available information.
2. Determine how knowing the new instance and its classification it can be used to improve the accuracy of the model.

The first of these functions is a classification task, while the second can be viewed as a supervised learning task [1].

3 Representation

In the proposed model, the case memory (CB) is represented by a set of categories $\{C_1, \cdots, C_n\}$ which are not necessarily disjoint. A *category* C_j is represented by a set of exemplars $\{e_{1j}, \cdots, e_{mj}\}$. An exemplar e_{ij}, which depicts a set of "similar" cases, is represented by a set of features $\{f_a, \cdots, f_g\}$. A *case* c_{kij} is represented by a set of features $\{f_1, \cdots, f_p\}$ and a *feature* is a term that represents a binary variable.

The model uses two *Bayesian networks* [7] as shown Fig. 2.

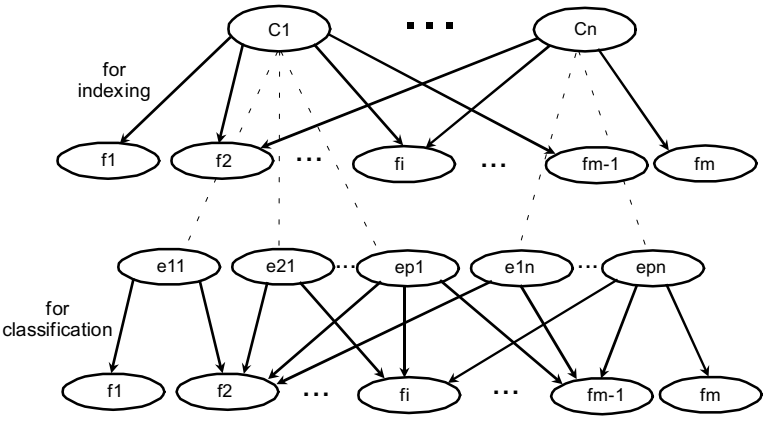


Fig. 2. Bayesian networks for indexing and classification.

The first Bayesian network (BN) is used as an index structure to categories. The second BN defines the relationships between the features and the exemplars. These networks are used in two stages as follows:

1. Given the features of a new case, the first network is used to determine the probability of each category $P(C_j|nc)$. This information ranks the order in which the categories can be searched for a similar exemplar.
2. Each category can then be investigated, in order of rank, to find an exemplar that is "similar" to a new case. That is, within a selected category, the second network is used to calculate the probability of each exemplar given the features of the new case. Thus, "similarity" is interpreted as the probability that an exemplar represents a case.

The decision of when to stop investigating is normally application dependent. A user may want to stop as soon as an exemplar with a *high* probability (e.g., above 0.9) is found, or the user may prefer to obtain all the exemplars with a probability above a threshold value.

To illustrate the representation, suppose we continue with the example of Fig. 1. Suppose each category is defined as shown below. The numbers in the exemplars indicate the actual cases that have been classified (or are represented) by the exemplar and the numbers in the features indicate the frequency of the feature in the exemplar. So, for example, e_1 is known to represent four actual cases and the feature f_1 occurs three times. Notice that the exemplar e_3 is in both categories.

category	exemplars	features
A	e_1 (4)	f_1 (3), f_2 (4), and f_3 (2)
	e_2 (2)	f_2 (2), f_4 (2), and f_5 (1)
	e_3 (3)	f_2 (1), f_4 (2), f_6 (3), and f_7 (2)
B	e_3 (3)	f_2 (1), f_4 (2), f_6 (3), and f_7 (2)
	e_4 (2)	f_2 (2), f_7 (2), and f_8 (1)

Figure 3 shows the two Bayesian networks for this situation, where the numbers labelling the arcs represent the number of times the features are true in a category (top network) or exemplar (bottom network). These frequencies are used to compute the conditional probability of each feature given its categories or exemplars. The number of cases within each category or exemplar are used to obtain the prior probabilities of the categories or exemplars in the Bayesian networks.

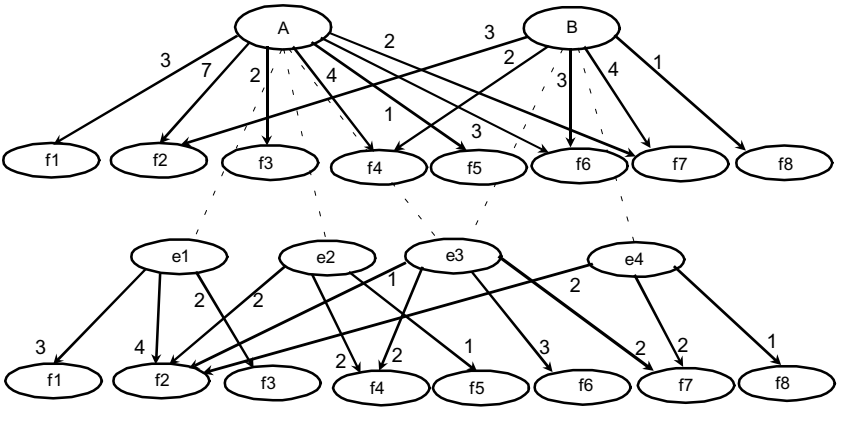


Fig. 3. The organisation structure.

4 Classification

This section presents the classification algorithm, figure 4. For simplicity and conciseness, we present a version that stops as soon as an exemplar is found whose conditional probability is above a user defined threshold.

$Classify(nc)$

Input: A new case described by a set of features $nc = \{f_a, \cdots, f_i\}$
Results: the exemplar e_c that best classifies the new case and
the category list CL that classify e_c

The following local variables are used:
H is a list sorted of categories
E is a list sorted of exemplars
C_c is the current category
$done$ is a boolean variable

Step 1. Rank the Categories

Compute conditional probabilities in the category-features Bayesian network

for each $C_j \in casebase$ do
compute $P(C_j|nc)$
set H to the list of categories ranked in descending order of $P(C_j|nc)$

Step 2. Determination of an Exemplar

$e_c = nil$
$C_c = first(H)$ The list H returns $\emptyset$ at the end
$done = false$

while (not $done$) and ($C_c \neq \emptyset$) do begin

In the exemplar-features Bayesian network

for each $e_i \in C_c$ do
compute $P(e_i|nc)$
set E to the list of exemplars in C_c ranked in descending order of $P(e_i|nc)$

$e_c = first(E)$
if $P(e_c|nc) > threshold$ then
$done = true$
else
$C_c = next(H)$
end(if)
end(while)

if done then
$CL =$ all categories that contain (e_c)
else
$e_c = nil$
$CL = \emptyset$
end(if)

output (e_c, CL)

Fig. 4. Classification algorithm

The first step of the algorithm computes the conditional probabilities of the categories given the features. This computation is carried out by propagation techniques developed by Lauritzen and Spiegelhalter [8]. These probabilities are then used to rank the categories. The second step then searches through the ranked list of categories for the first exemplar whose conditional probability is above the threshold. Then the exemplar confirms the hypothesis and the search process finishes.

5 Learning

Figure 5 shows the learning algorithm. It uses a training case and its categories[1] in an attempt to improve the accuracy of the model.

Input: A training case described by a set of features $nc = \{f_a, \cdots, f_i\}$ and
a set of categories $L = (C_o, \cdots, C_p)$ to which it *jointly* belongs.
Results: Updating model for CBR.

1. $Classify(nc)$

The classification output are stored in the following local variables:
CL is a list of categories that classify the nc
e_c is the exemplar that best classifies the nc

2. if $(CL = \emptyset)$ then
$e_c = nc$
for each category $C_k \in L$ do
add_exemplar(e_c, C_k)
return
end(if)

/* Learning */
3. if $L = CL$ then
for joint category $jC \in L$ do
$pe_c = prototypicality(e_c, jC)$
$pnc = prototypicality(nc, jC)$
if $(pnc > pe_c)$ then
nc replaces e_c in the definition of jC
else
$e_c = nc$
for each category $C_k \in L$ do
add_exemplar(e_c, C_k)
end(if)

Fig. 5. Learning algorithm

The learning procedure works as follows. First it obtains the behaviour of the current model by using the classification algorithm to return the categories which would jointly classify the training example. If no categories are found (i.e. $CL = \emptyset$), then the training case is added as an exemplar to all its categories[2].

[1] An exemplar can be in several categories.
[2] This may also involve creating categories.

If classifying does return a list of categories, there are two situations. First, if the returned list is not the same as the training case's categories, then the case is added as an exemplar to each of its categories. Notice, that the test is for equivalence since the training case's categories are viewed as a joint category. Alternatively, if the predicted list of categories corresponds to the actual joint categories, then we have to consider whether to retain the training instance as an exemplar, and whether it should replace the exemplar that was used to classify it. In order to decide if the new case is a better exemplar than the exemplar that classifies it, the definition of a prototype is used. A prototype is a representative case that denotes a set of cases bound together by *family resemblance* [13]. So, the family resemblance between two cases depends on the number of features that they have in common. In other words, it is a similarity measure between them.

In the situation under consideration, a prototypical exemplar will be one that (i) best represents its similar cases in the category, and (ii) has least similarity with the cases represented by other exemplars.

The first of these considerations is known as *focality* and the second is known as *peripherality* [4]. Before describing the measures used for focality and peripherality, the concept of a *summary representation* of an exemplar is defined

In the proposed model, the cases represented by an exemplar are not stored. Instead, only a *summary representation* [14] (SR) of the similar cases is stored. This summary representation is an extension of the Bayesian network that represents the exemplars of the category. It includes additional nodes and arcs to describe features that are not present in the exemplar, but present in some cases it represents.

For example, supose the exemplar e_3 represents the cases c_6 and c_7 as shown in Table 1. Then, the summary representation of e_3 will include all the features of the cases c_6 and c_7 as shown in Figure 6.

Table 1. Similar cases with e_3

case	features	prototype
c_3	$f_2 f_4 f_6 f_7$	e_3
c_6	$f_4 f_6 f_9$	
c_7	$f_6 f_7$	

Given such summary representations, the focality and peripherality measures can now be defined. The focality measure of an exemplar e_i in a category C is defined as follows.

$$Focality(e_i, C) = P(SR_{e_i}|e_i)$$

Where, $P(SR_{e_i}|e_i)$ measures how well the exemplar e_i represents its cases.

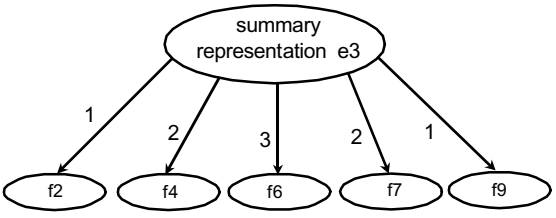


Fig. 6. Summary representation of e_3

The peripherality measure of an exemplar e_i in a category C is defined as follows.

$$Peripherality(e_i, C) = \frac{1}{k} \sum_{j=1}^{k} P(SR_{e_j}|e_i) \ \forall j \neq i \in C$$

Where, $P(SR_{e_j}|e_i)$ determines the similarity of e_i with respect to other exemplars e_j. Thus, if this peripherality measure is large than the exemplar e_i will be similar to other exemplars and therefore not peripheral. If it is small, it can be expected to represent cases that are not covered by the other exemplars.

In general, an examplar is considered good if it represents a set of cases that are not already represented by other exemplars. Hence the prototypicality measure of an exemplar e_i in a category C is defined as the diference between its focality and its peripherality:

$$Protypicality(e_i, C) = Focality(e_i, C) - Peripherality(e_i, C)$$

Now, returning to the earlier example, suppose that we have the following new training cases: $c_{12} = \{f_4, f_6, f_7, f_9\}$ and it is correctly classified by exemplar e3. Then, since c_{12} gives greater prototypicality, the model changes and c_{12} becomes the exemplar e_{12} instead of e_3.

6 Experimental Results

A simplified version of the model [12] has been implemented and evaluated experimientally on three datasets, votes zoo, and audiology, available from the Univeristy of California repository of datasets. Due to the lack of space, this section presents only a very brief summary of the results which are presented more fully in [12]. Table 2 summarises the characteristics of each dataset.

Each experiment randomly partitioned the data into a 70% training set and a 30% testing set. and was repeated 20 times to obtain an average accuracy and compression ratio (defined as the proportion of cases not retained) for each class. Tables 3 and 4 give the results for the votes and the zoo datasets. As the results show, the model performs well both in terms of accuracy and the number of exemplars retained. The overall compression ratio for the votes dataset is 97%

Table 2. A summary of the datasets.

Dataset name	Number of cases	Number of features	Values in features	Number of concepts	Missing values
Votes	435	16	2	2	Y
Zoo	101	16	2	7	N
Audiology	226	69	2	24	Y

Table 3. Averages results for the votes dataset.

No.	Category	Training cases	Testing cases	Exemplars	Accuracy ±95% conf. int.
1	Republicans	119.20	47.8	2.1	96%±1.9%
2	Democrats	185.05	81.95	4	84%±2.0%

with an accuracy of 89%. The overall compression ratio for the zoo dataset is 87% with an accuracy of 92%.

For the audiology dataset, the accuracies obtained are good for some of the categories and poor for some categories where there are few training cases. This behaviour is to be expected since the model is not expected to learn exemplars from a few cases. The motivation for testing the model proposed with audiology dataset was to enable some comparison with a closely related system, PROTOS [3] that utilised that data. Although it would be incorrect to draw comparative conclusions from the results of the single trial presented in [3], it is encouraging that the results obtained in terms of accuracy and compression ratio, are similar to those obtained when PROTOS was trained with the aid of an audiology expert (see [12] for details).

Table 4. Averages results for the zoo dataset.

No.	Category	Training cases	Testing cases	Exemplars	Accuracy ±95% conf. int.
1	class-1	28.4	12.6	1.35	98%±1.7%
2	class-2	14.16	5.35	1	99%±1.4%
3	class-3	3.45	1.55	1.65	16%±12.3%
4	class-4	9.3	3.7	1	100%±0%
5	class-5	2.9	1.1	1	77%±19.3%
6	class-6	5.35	2.65	1	100%±9.8%
7	class-7	7.3	2.7	1.75	80%±9.6%

7 Model Extensions

The proposed model, as explained in section 3, uses two Bayesian networks, one for indexing categories and the other for representing exemplars. The structure of both Bayesian networks considers two important simplifications: (i) all the features are binary, and (ii) all the features are independent given the class/exemplar. In this section we outline how these assumptions can be relaxed.

7.1 Multilevel Features

Although a N-valued attribute can be represented by $N+1$ binary features, this approach produces two unwanted consequences: (a) the complexity of the model increases, (b) the binary features that represent a multivalued attribute are not conditionally independent.

A solution is to simply incorporate multivalued features in both Bayesian networks, for indexing and classification. However, there are some considerations from the CBR viewpoint that need to be considered. If there are multivalued attributes, two situations can arise:

1. A feature that corresponds to a value of the attribute is present, and this value is instantiated in the BN.
2. None of the values in the attribute is present. In this situation there are two alternatives: (i) leave the attribute unspecified (unknown) or (ii) add an additional value, "unknown", to each attribute, and instantiate this value.

Which of the previous two alternatives is better depends on the domain; in particular if it is important to distinguish ignorance (the attribute is not known for the case) and irrelevance (the attribute can be obtained but is not relevant for the case).

7.2 Dependent Features

The initial model assumes that all the feature variables are conditionally independent given the category or exemplar. Although this assumption simplifies the model and it is true in some cases, it is not generally satisfied in reality. Dependencies between features that are not taken into account can degrade the performance of the model.

Determining if two features are dependent can be done by measuring the mutual information or correlation between each pair of features given the class/exemplar. If these are above certain threshold, then, it can be considered that the features are dependent; otherwise they are conditionally independent.

8 Related Work

The proposed model uses Bayesian networks for incremental exemplar-based classification. The related work can therefore be classified in terms of: (i) inductive models with supervised and unsupervised learning, (ii) case-based and exemplar-based models, and (iii) use of Bayesian models in CBR.

Each of these areas has been extensively covered in the literature. The work on inductive models includes *instance-based learning* (IBL) algorithms [1], and explanation based learning [9]. The work on exemplar-based models includes that of Biberman, who examined the role of prototypicality [4], and the work on Protos [3]. The use of Bayesian networks in CBR is increasing rapidly and includes the work of Aha and Chang [2], Myllymäki and Tirri [10], Breese and Heckerman [5], Chang and Harrison [6]. A detailed analysis of the main differences between the proposed model and the above models is presented in [12].

Protos is perhaps the best known exemplar-based system [3]. It is available freely for experimental purposes and has therefore been well studied. In terms of the stages, Protos is quite similar to the proposed model. However, there are three significant differences. First, in Protos, an exemplar is represented by a prototypical case while in the proposed model, an exemplar is represented by a Bayesian network, where the random variables of the network are determined by the prototypical case. Second, the proposed model uses Bayesian networks in the retrieval process instead of heuristics for setting up and using indices (e.g., remindings, difference links, censors) thereby providing a better foundation for the proposed model. Thirdly, in the learning phase, Protos learns the importance of its indices by explanation, while the proposed model learns directly from the data. Also, Protos retains those cases that are not correctly classified as new exemplars. Cases that are correctly classified result in an increase of an exemplar's prototypicality but are discarded. In contrast, the proposed model attempts to use the notion of prototypicality to determine if a new case, that is correctly classified, would make a better exemplar.

9 Conclusion

This paper proposes an exemplar-based model with foundations in Bayesian networks. The model utilises two networks: one for ranking categories, and another for identifying exemplars within categories. This view of exemplar-based models leads to the notion of assessing similarity by conditional probabilities. Thus in classification, one can obtain the probability of an exemplar given the features of the new case.

Learning is incrementally conducted each time a new case is classified. When a new training case is well classified the model determines whether the new case will make a better exemplar than the one used to classify it. This is achieved by utilising the notions of prototypicality, focality, and peripherality. When a new case is not properly classified, it is simply added to all the categories to which it belongs. As future work we plan to implement and test the proposed extensions for modelling multivalued and dependent features.

References

1. D.W. Aha. Case-based learning algorithms. In *Proc. of the DARPA Case-Based Reasoning Workshop*, pages 147–158, Washington, DC, U.S.A.: Morgan Kaufmann, 1991.

2. D.W. Aha and L.W. Chang. Cooperative bayesian and case-based reasoning for solving multiagent planning tasks. Technical Report AIC-96-005, Navy Center for Applied Research in AI Naval Research Laboratory, Washington, DC, U.S.A., 1996.
3. R. Bareiss. *Exemplar-based knowledge acquisition. A unified approach to concept representation, classification, and learning.* Academic Press Inc., Harcourt Brace Jovanovich Publishers, San Diego, CA, U.S.A., 1989.
4. Y. Biberman. The role of prototypicality in exemplar-based learning. In Nada Lavrač and Stefan Wrobel, editors, *Proc. of Machine Learning: ECML-95, 8th European Conference on Machine Learning*, pages 77–91, Heraclion, Crete, Greece, 1995.
5. J.S. Breese and D. Heckerman. Decision-theoretic case-based reasoning. In *Proc. of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 56–63, Ft. Lauderdale, U.S.A., 1995.
6. L. Chang and P. Harrison. A case-based reasoning testbed for experiments in adaptive memory retrieval and indexing. In D.H. Aha and A. Ram, editors, *Proc. of the AAAI fall Symposium on Adaptation of Knowledge for Reuse*, Menlo Park, CA, U.S.A.: AAAI Press., 1995.
7. T. Dean, J. Allen, and Y. Aloimonos. *Artificial Intelligence theory and practice.* The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, U.S.A., 1995.
8. S. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society series B*, 50(2):157–224, 1988.
9. S. Minton, J.G. Carbonell, C.A. Knoblock, D.R. Kuokka, O. Etzioni, and Y. Gil. Explanation-based learning: a problem solving perspective. In Jaime Carbonell, editor, *Machine Learning: Paradigms and Methods*, pages 63–118. Cambridge, MA, U.S.A.: MIT/Elsevier Science, 1990.
10. P. Myllymäki and H. Tirri. Massively parallel case-based reasoning with probabilistic similarity metrics. In Klauss-Dieter Althoff Stefan Wess and Michael M. Ritcher, editors, *Topics in Case-Based Reasoning*, pages 144–154. Volume 837, Lecture Notes in Artificial Intelligence. Springer Verlag, 1994.
11. B.W. Porter, R. Bareiss, and R.C. Holte. Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence, University of Texas, Austin Texas, U.S.A.*, (45):229–263, 1990.
12. Andrés F. Rodríguez. *A probabilistic exemplar based model.* PhD dissertation, (1998), Department of Computer and Mathematical Science, TIME Research Institute, University of Salford, Salford, England, 1998.
13. E. Rosch and C.B. Mervis. Family resemblance studies in the internal structure of categories. *Cognitive Psychology*, (7):573–605, 1975.
14. E. Smith and D. Medin. *Categories and concepts.* Cambride: Harvard University Press, U.S.A., 1981.

Intensification and Diversification Strategies with Tabu Search: One-Machine Problem with Weighted Tardiness Objective

Ricardo P. Beausoleil

Institute of Cybernetics Mathematics and Physics
Department of Optimization, City of Havana, Havana
p.c. 10400,Cuba Fax: 537 333373
rbeausol@cidet.icmf.inf.cu

Abstract. This article investigates several intensification and diversification strategies for the one machine problem with weighted tardiness objective. The aim of this study was to achieve a balance between intensification and diversification strategies. The use of intensification by decomposition, path relinking, frequency-based memory and large step optimization in intensification and diversification process, are combined in several procedures. We perform several computational experiments to study the effect of several combination of these strategies. Our result indicates that combined "large step optimization" with "intensification-diversification approach" and adicional intensification with "path relinking" achieve the betters performance.

Keywords: Adaptive approach, Tabu Search, scheduling problems, weighted tardiness

1 Introduction

Tabu Search is a meta-heuristic that makes use of historical information. The historical information is kept in three kinds of memory functions: the short-term,the intermediate and long term memory function. Short term memory function has the task to memorize certain complete solutions or attributes of the search process of the recent past and it is incorporated in the search via one or more tabu list (recency-based memory). Frequency-based memory is fundamental to longer term considerations. Frequency-based memory provides a type of information that complements the information provided by recency-based memory.

Two highly important components of tabu search are intensification and diversification strategies. Intensification strategies are based on modifying choice rules to encourage move combinations and solution features historically found good.

Intensification strategies generate "neighbors" by either grafting together components of good solutions or by using modified evaluations strategies that favor the introduction of such components into the current (evolving) solution.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 52–62, 2000.

The diversification strategies leads the search process to examine unvisited regions. The objective of a diversification is to produce a new starting point, wherefore the local search continue.

In the approach to obtain a new starting point, two kind of diversifying strategies can be distinguished, strategies using a restart method and strategies using a method which lead the search to a new regions in an iterative fashion, this kinds of diversification methods usually provide advantages over restart methods, [8].

In our study we have implemented a diversification as an iterative method, using frequency counts, and a large-step procedure. Large step optimization consist of three main routines, a large-step phase, a small-step phase and an accept/reject test, which all three are consecutively executed for certain number of large step iterations. This procedure should make a large modification in the current solution to drive the search to a new region and, at the same time, to perform some kind of optimization to obtain a solution not too far away from a local (or global) optimal schedule.

Additional search intensification using path relinking, a strategy proposed in connection with tabu search, which has been rarely used in actual implementation.One of the few path relinking implementations appears in Laguna and Martí [1]. Path Relinking offers a useful integration of intensification and diversification strategies. This approach generates new solutions by exploring trajectories that connect elite solutions – by starting from one of these solutions, called an initiating solution, and generating a path in neighborhood space that leads toward the other solutions, called guiding solutions.

2 A Case of Study

The problem to be discussed in this paper is the scheduling of n jobs on a single machine where the goal is to minimize the total weighted tardiness. One-Machine is a productive capability able to handle exactly one job or operation, the only decisions to be made are sequencing and release timing of jobs or operations from the input queue (we have a single input queue) to the resource.

We consider the machine processes jobs one at a time serially, the resource is available over the scheduling interval t_s to t_e, n-single operations jobs arrive over the interval, the job i has a processing time p_i, a ready time r_i, a due date d_i, preemption model: nonpreemtive, objective function: total weighted tardiness.

The lateness of the job i is the amount of time (positive or negative) by which the completion of activity i exceeds its due date, $L_i = C_i - d_i$. Tardiness is defined as rectified lateness, i.e; the lateness of job i if it is positive; if the lateness of job i is not positive, the tardiness is zero: $T_i = max\{0, L_i\}$, tardiness, reflects the fact, in many situations, distinct penalties and other costs will be associated with positive lateness, but no penalties or benefits will be associated with negative lateness. Minimum weighted tardiness $\sum_i w_{T_i} Ti$ is a good objective, but problems using this objective are very difficult to solve exactly.The static form of this problems was first presented by McNaughton [2].In this problem we need only consider permutations of the n jobs to find the optimal schedule.However,

even for modest-size problem, complete enumeration is not computationally feasible since it requires the evaluation of n! sequences.

A number of approaches have been suggested for solving this problem without performing an explicit enumeration.Held and Karp [3] and Lawler [4] present dynamic programming formulations which require the consideration of 2^n possible subsets of the n jobs. Although much more efficient than complete enumeration, this method is still computationally infeasible for modest-sized problems (e.g. a 20-job problem requires the storage of more than a million numbers).More recently several techniques were developed, OPT-like rules came in the late 1970s and early 1980s,while bottleneck dynamics started in the early 1980s.Similarly developed a number of newer mathematical methods: beam search, simulated annealing, genetic algorithms, tabu search and others. In this paper presents three heuristics, based in TS methodology to solve the weighted tardiness problem for one-machine scheduling.

3 Adaptive Approach Heuristics

3.1 Neighborhoods

Two methods of generating neighborhoods are presented in this paper, an adjacent pairwise interchange operations where two adjacent jobs are interchange and a general pairwise interchange operation that considers the neighborhood generated by every possible pairwise interchange. A general three-move to the left or right is used in order to incorporate a perturbation move.

3.2 A Tabu Search Framework

TS was proposed in its present form by Glover[5]. The basic idea of TS have also been sketched by Hansen [9]. Tabu Search can be described as an intelligent search that uses adaptive memory and responsive exploration. The adaptive memory feature of TS allows the implementation of procedures that are capable of searching the solution space economically and effectively. Responsive exploration is based from the supposition that a bad strategic choice can yield more information than a good random choice.

The memory structures in tabu search operate by reference to four principal dimensions, consisting of recency, frequency, quality, and influence. In our implementation we use recency-based memory, frequency-based memory and the quality to differentiate the merit of the solution visited during the search.

Tabu Search (TS) can be viewed as an iterative technique which explores a set of problem solutions, denoted by X, by repeatedly making moves from one solution s to another solution $s^{'}$ located in the neighborhood $N(s)$ of s Glover[1], then in order to derive a TS, it is necessary to define a neighborhood structure, that is, a function that associates a set of solutions $N(s)$ with each solution s.

Here, we present the general framework of ours tabu search method implemented, while details will be discussed below.

Procedure TS-LS;
{ repeat
  repeat
    generate an initial solution s in EDD order;
    set tabulist empty;
    generate $N^* : N^*(s) = \{s^{'} | s^{'} \in N(s)|$satisfies the global aspiration
    or not tabu and N(s) is generated via adjacent pairwise interchange$\}$;
  until achieve the threshold for the amount of non-improving move;
  execute a perturbation move via general three-way interchange ;
until iter = numiter;
}

Procedure TS-ID;
{ repeat
  repeat
    generate an initial solution s in EDD order;
    set tabulist empty;
    generate N^*:$N^*(s) = \{s^{'} | s^{'} \in N(s)$:satisfies the global aspiration
    or not tabu and N(s) is generated via adjacent pairwise interchange$\}$;
    update the frequency-based memory;
  until achieve the threshold for the amount of non-improving move;
  update the residence measure for each one
  of the attributes of elite solutions;
update the penalty in order to implement an
intensification strategy for elite solution
  repeat;
    generate $N^* : N^*(s) = \{s^{'} : s^{'} \in N(s)|$ satisfies the global aspiration
    or not (tabu or penalizing move) and $N(s)$ is generated via
    general pairwise interchange$\}$;
  until achieve the threshold for the amount of non-improving move;
  diver=true(*diver is an indicator of diversification*);
until iter=numiter;
}

3.3 Memory Structures

The main components of tabu search algorithms are memory structures, in order to have a trace of the evolution of the search. TS maintains a selective history of the states encountered during the search, and replaces $N(s)$ by a modified neighborhood $N^*(s)$. In our implementation we use recency and frequency based memories. Associate with these memories exist a fundamental memory structure so called tabu list, in our case based in an attribute memory, that is, we take account the identity of the pair of elements that change positions. In order to maintain the tabu list, we use one array $tabuend(e)$, where e ranges our attributes (see Glover and Laguna,[7]), in our case position of jobs.

Our implementation uses a complementary tabu memory structure, that is, associated with the array *tabuend* we have the array $frequencycount$, it show the distribution of the move attributes, i.e, the numbers of times each positions of jobs has been exchanged. We use these counts to diversify the search. Our algorithm uses the frequency information to penalize non-improving move by assigning a penalty to swaps of position pairs with greater frequency.

This implementation uses a tabu restriction that forbids swaps based in one array that records a tabu tenure for each position of jobs separately, making the move tabu inactive, if one of the selected attribute is tabu inactive. This implementation increase the percentage of available move that receive the tabu classification, permitting to have an small tabu tenure, the length of tabutenure = 3, and can be implemented with much less memory.

3.4 Aspiration Criteria

Aspiration criteria are introduced in tabu search to determine when tabu restriction can be overridden, thus removing a tabu classification. In our tabu search implementation we use a global form of aspiration by objective, that is, a move aspiration is satisfied, permitting s^{trial} to be a candidate for selection, if $c(s^{trial}) < bestcost$.

3.5 Diversification and Intensification

The diversification stage encourages the search process examine unvisited regions and to generate solutions that differ in various significant ways from those seen before. Intensification strategies are based on modifying rules to encourage move combinations and solution features historically found good.

In order to implement these strategies in ours TS procedures, we conceive frequency mesure that consist of the numbers of times that the jobs occupies every one of the positions in a goods schedules, this is a so called recency frequency. In a case of an intensification strategy a high residence frequency, may indicate that one attribute is highly attractive if we obtain the frequency of the sequence of high quality solutions, and in the diversification we can use this as certain tabu status to lead the process to new regions of the solutions space.

We implement two approaches, the first approach uses two sequence phases, one to diversification and the second phase to intensification, in this phase the tabulist and frequencycount are initialized, in the second approach the intensification phase is included within the diversification phase and the tabulist and frequencycount are not initialized and the value of tabu tenure, for the intensification, is equal 2. The diversification phase termites, when an improving move is reached.

Our intensification approach is an intensification by decomposition, where restriction are imposed on part of the solution structure in order to allow a more concentrated focus on the other parts of the structure. In our case, the jobs pertaining to intersection of elite sequence are highly penalized and a general pairwise interchange is applied.

3.6 Large-Step Optimization

In the procedure TS-LS we combine the large-step optimization with a tabu search approach, that was applied to a job shop scheduling problem by H.Ramalhinho [10]. The large-step optimization consists of tree main routines: a large-step phase, a small-step phase, and an accept/reject test, which all tree are consecutively executed for a certain number of large iterations.

An important part of the large-step optimization methods is the large-step phase, that make a large modification in the current solution to drive the search to a new region.

Here we use a general 3-way interchange to left or right as the large step phase and, as small step a simple tabu search approach.

3.7 TS Pseudo Code

Here formally we describe two of ours tabu search approaches, the TS-DI is identically to TS-ID, the difference is that, the phase of intensification is after the diversification.

Tabu Search approach with Large step.

procedure TS-LS;
{ $iter = 0$;
generate starting solution in EDD order;
repeat;
  tabulist empty;
  $freqcount = 0$;
  $bestcost = best\, c(s^{now})$;
  repeat;
    $iter = iter + 1$;
    $counter = 0$;
    repeat;
      $counter = counter + 1$;
      generate a move via adjacent pairwise interchange;
      select(i,j| $tabuend[i] < iter\, or\, tabuend[j] < iter$
            or $c(s^{trial}) < bestcost$);
      if non-improving move penalize $c(s^{trial})$;
      choice the best s^{trial};
    until $counter = n - 1$;
    $s^{now} = best\, s^{trial}$;
    $c(s^{now}) = c(best\, s^{trial})$;
    if improving move
      $bestcost = c(s^{now})$;
      $s^{best} = s^{now}$;
    $movei = i; movej = j$;
    $freqcount[movei] = freqcount[movei] + 1$;
    $freqcount[movej] = freqcount[movej] + 1$;

$tabutenure[movei] = tabutenure[movei] + iter;$
$tabutenure[movej] = tabutenure[movej] + iter;$
until $nonimprove = 20;$
generate a perturbation move via general 3-way
interchange to left or right;
until $iter = 30;$

Tabu Search approach with Intensification-diversification.

procedure TS-ID;{
$iter = 0;$
tabulist empty;
$freqcount = 0;$
generate starting solution in EDD order;
repeat;
$iter = iter + 1;$
$bestcost = best\, c(s^{now});$
repeat;
$counter = 0; nonimprove = 0$
repeat;
$counter = counter + 1;$
generate a move via adjacent pairwise interchange;
select(i,j| if diver then ($tabuend[i] < iter\, or\, tabuend[j] < iter$)
and $s^{trial}[i] \neq s - to[i]\, and\, s^{trial}[j] \neq s - to[j];$
or $c(s^{trial}) < bestcost$)
otherwise $tabuend[i] < iter\, or\, tabuend[j] < iter$
or $c(s^{trial}) < bestcost$));
if non-improving move penalize $c(s^{trial});$
choice the best $s^{trial};$
until $counter = n - 1;$
$s^{now} = best\, s^{trial};$
$c(s^{now}) = c(best\, s^{trial});$
if improving move
then {
diver=false;
Intensification(s^{now});
update $bestcost\, and\, s^{best};$
}
$movei = i; movej = j;$
$freqcount[movei] = freqcount[movei] + 1;$
$freqcount[movej] = freqcount[movej] + 1;$
$tabutenure[movei] = tabutenure[movei] + iter;$
$tabutenure[movej] = tabutenure[movej] + iter;$
until $nonimprove = 20;$
$s^{now} = best\, s^{trial};$
$c(s^{now}) = c(best\, s^{trial});$

if improving move then $freqconfg[s^{best}] = freqconfg[s^{best}] + 1$;
record in s-to[i] the job that more has occupied the
position i in the elite solutions found during the search;
diver=true;(***Diversification phase***)
until $iter = 30$;

4 Computational Experiment

The first experiment was to explore the effect over the neighborhoods of changing tabutenure from 3 to 5 and 7. The experiment revealed that the greater number of neighbor without repetition was reached when tabutenure is set to 3.

The second experiment has the goal of finding an appropriate value for the number of non improving move nonimprove in the phase of diversification and intensification, and the value of the parameter iter. For this purpose, was performed the experiments with random instances of size 20, 50, 100 jobs.

The results in the tables 1, 2, and 3 shows in the first column the numbers of jobs in each instance. The second column show the maximum tardiness reached in ten runs for each instance. The third column indicates the best Tardiness in these ten runs. The fourth column shows the average iter where occurred the last improving move. The fifth column indicates the average solution time.

We use a random generater with uniform distribution for the problems.

Ours experiments shows that the approach with Intensification within Diversification have betters solutions, otherwise the approach Large Step have betters times, this suggest the idea of integrate the Large step approach with the Intensification-Diversification approach. The tables 4 and 5 shows experiments with TS-ID and Tabu with Large Step, Intensification-Diversification phases called TS-LSID.

In this approach the Large Step Optimization is used to get a good solution very quickly and then improve this solution with Intensification Diversification strategy.

Table 1. Summary of results TS-LS.

N	MaxTard	MinTard	Iter	Seconds
20	1258	1220	11	1
	993	970	9	1
	993	970	11	1
	698	685	10	1
50	11164	10941	3	17
	10664	10344	3	17
	10153	9995	3	16
	10077	9938	2	18
100	44283	44246	1	181
	42598	42593	1	165
	44910	44610	4	203
	44523	43463	1	195

Table 2. Summary of results TS-DI.

N	MaxTard	MinTard	Iter	Seconds
20	1255	1217	13	1
	1977	1966	12	1
	981	964	9	1
	698	685	10	1
50	11087	10858	16	20
	10398	10319	15	20
	9985	9957	16	19
	10004	9864	19	20
100	44210	44139	19	211
	42516	42459	18	231
	44594	44518	19	227
	43507	43372	20	227

Table 3. Summary of results TS-ID.

N	MaxTard	MinTard	Iter	Seconds
	1244	1215	13	1
	1975	1964	7	1
20	975	965	5	1
	699	685	8	1
	10936	10835	12	20
	10559	10306	11	22
50	9980	9947	12	19
	9957	9866	17	21
	44283	44122	20	190
	42498	42459	22	229
100	44813	44493	16	209
	43572	43383	18	227

Table 4. Summary of results TS-ID.

N	MaxTard	MinTard	Iter	Seconds
	953	921	14	1
	515	477	14	1
20	998	974	8	1
	1103	1085	13	1
	8853	8675	13	23
	10313	10276	15	19
50	8483	8418	14	22
	9384	9190	16	21
	39677	39527	20	203
	44117	44011	15	231
100	44687	44625	20	212
	33163	32975	18	234

Table 5. Summary of results TS-LSID.

N	MaxTard	MinTard	Iter	Seconds
	928	921	9	1
	506	476	14	2
20	1015	973	12	2
	1102	1083	14	1
	8841	8672	15	21
	10440	10274	15	20
50	8476	8420	11	22
	9219	9188	14	23
	39628	39500	22	204
	44224	44016	18	225
100	44649	44613	20	240
	33070	32989	20	240

5 Intensification with Path Relinking

Path Relinking has been suggested as an approach to integrate intensification and diversification strategies [8]. This approach generates new solutions by exploring trajectories that "connect" high-quality solutions – by starting from one of the solutions, called an initiating solutions, and generating a path in neighborhood space that leads toward the others solutions, called guiding solutions. In our implementation, the path relinking strategy is used with the goal of strengthening the intensification phase. During path relinking, the main goal is to incorporate attributes of guiding solution while at the same time recording the objective function values. In the current development, the procedure stores a small set of the elite solutions to be used as reference point (the best three solutions found during the search). The relinking process implement in our search may be summarized as follows:

The set of elite solutions is constructed during the intensification and diversification phases (the best three solutions).

Designate a subset of the best solutions to be reference solutions (three different subsets are generated, all combinations of two solutions) taking as guide solutions the best in terms of the objective function and generating two trajectories to the best solutions.

Incorporate attributes of guiding solutions by inserting and exchanging moves. In our implementation s is a permutation of numbers of jobs, that is, $s = (s_1, s_2, \ldots, s_n)$ then, we define $Insert(s_j, i)$ to consist of deleting s_j from its current position j to be inserted in position i.

$$\begin{aligned} s^{'} &= (s_1, \ldots, s_{i-1}, s_j, s_i, \ldots, s_{j-1}, s_{j+1}, \ldots, s_n) \text{ for } i < j, \\ &= (s_1, \ldots, s_{j-1}, s_{j+1}, \ldots, s_i, s_j, s_{i+1}, \ldots, s_n) \text{ for } i > j. \end{aligned}$$

and $Exchange(s_i, s_j)$ as was described in pairwise interchange, that is:

$$s^{'} = (s_1, \ldots, s_{i-1}, s_j, s_{i+1}, \ldots, s_{j-1}, , s_i, s_{j+1}, \ldots, s_n).$$

Neighborhood to Path Relinking: two neighborhood were considered to our path relinking, if i is the position that s_j occupies in the guide solution, then

$$\begin{aligned} N_1 &= \{s^{'} : Insert(s_j, i), i, j \in \{1, 2, \ldots, n\}\} \\ N_2 &= \{s^{'} : Exchange(s_i, s_j), i, j \in \{1, 2, \ldots, n\}\}. \end{aligned}$$

In conjunction with these neighborhood, one strategy is defined to select the best move as follow:

$$S^* = bestmove(E, I, xbest) \text{ in terms of our objective function}$$

where E is the exchange operation and I is the insertion operation and xbest is the last best solution.

The table 6 shows the results of tabu search with large-step, intensification-diversification (TS-LSID) versus TS-LSID with additional path relinking (TS-PR).

Table 6. Summary of results TS-LSID versus TS-PR.

jobs	*TS-LSID*		*TS-PR*		*CPU*
N	MaxTard	MinTard	MaxTard	MinTard	Seconds
20	502	494	499	492	1
	1125	1113	1122	1113	1
	671	654	666	654	1
	1062	1061	1062	1061	1
50	10801	10681	10801	10680	20
	9983	9879	9979	9879	20
	9303	9260	9303	9259	19
	7798	7722	7796	7722	20
100	44658	44627	44655	44626	241
	38666	38613	38665	38607	233
	43632	43465	43630	43465	230
	37639	37487	37639	37476	239

6 Conclusions

In this paper we have develop several tabu search procedures for the one machine dynamic problem with weighted tardiness objective. The performance of these problems have been proved using 36 random problems instance of several types every one ran ten times.

An important goal in our research is to prove that the strategies of intensification diversification play an important role to obtain a good and quickly solution. The Large step optimization seem a good strategy to obtain in this case a good initial solution.

Our additional intensification path relinking prove, that can be improved the solution without significative additional time.

References

1. M. Laguna and R. Martí (1997) "Grasp and Path Relinking for 2-Layer Straight Line Crossing Minimization", University of Colorado at Boulder.
2. R.McNaughton, Scheduling with Deadlines and Loss Functions, Management Science, (Sept.1959), Vol. 6, No. 1, pp.1–12.
3. M. Held and R. M. Karp, (Mach 1962), A Dynamic Programming Approach to Sequencing Problems, Journal of the Society for Industrial and Applied Mathematics, Vol. 10, No. 1, pp. 196–210.
4. E. Lawler, (Oct. 1964), On Scheduling Problems with Deferral Costs, Management Science, Vol. 11, No.2, pp. 280–288.
5. F. Glover,(1986), Future paths for integer programming and links to artificial intelligence, Computers and Ops. Res., 5, 533–549.
6. F. Glover, E.Taillard and D. de Werra, (1993) A user's guide to tabu search, Ann. Oper. Res. 41, 3–28.
7. F. Glover and M. Laguna, (1993), Modern Heuristic Techniques for Combinatorial Problems, published in the Halsted Press, John Wiley & Sons, Inc., chapter 3 , 70–147.
8. F. Glover, M. Laguna, T. Taillard and D. de Werra,(1993) Tabu Search, Annals of Operations Research, 41.
9. P. Hansen, (1986), The steepest ascent mildest descent heuristic for combinatorial programming, Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy.
10. H. Ramalhinho Lourenco and Michiel Zwijnenburg,(1998) Combining the Large-step Optimization with Tabu-Search: Application to The Job-Shop Scheduling Problem, Meta-Heuristics: Theory and Applications, Kluwer Academic Publishers, Ch 14, 219–235.
11. H. Emmons, (July-august 1969), One-Machine Sequencing to Minimize Certain Functions of Job Tardiness, The Journal of the Operations Research Society of America, Vol 17 Number 4 , 701–715.
12. J. Shwimer, (1972), On the N-Job, One-Machine, Sequence-Independent Scheduling Problem with Tardiness Penalties: A Branch-Bound Solution, Management Science, Vol. 18, No. 6 February, B-301.
13. T. E. Morton and D. W. Pentico, (1993), Heuristic Scheduling Systems, with Applications to Production Systems and Project Management, Wiley Series in Engineering and Technology Management.

A Methodology to Parallel the Temperature Cycle in Simulated Annealing

Héctor Sanvicente Sánchez[1] and Juan Frausto Solís[2]

[1] IMTA, Paseo Cuauhnahuac 8532, Col Progreso, C.P. 62550, Jiutepec, Morelos, México
hsanvice@tlaloc.imta.mx
[2] ITESM Campus Morelos, Reforma 182-A, Col. Lomas de Cuernavacca, A.P. 99-C, Cuernavaca, Morelos, México
jfrausto@campus.mor.itesm.mx

Abstract. Simulated Annealing (SA) is usually implemented in a sequential way. We propose a Methodology to Parallel the Simulated Annealing Algorithm (MPSA). This methodology carries out the parallelization of the cycle that controls the temperature in the algorithm. This approach lets a massive parallelization. The initial solution for each internal cycle may be set through a Monte Carlo random sampling to adjust the Boltzmann distribution at the cycle beginning. In MPSA the communication scheme and its implementation must be in an asynchronous way. Through a theoretical analysis we establish that any implementation of MPSA leads to a Simulated Annealing Parallel Algorithm (SAPA) that is in general more efficient than its sequential implementation version.

Keywords: Simulated Annealing, Combinatorial Optimization, Parallel Algorithms

1. Introduction

The Simulated Annealing Algorithm (SA) is a simple an effective optimization method to solve NP-hard problems [1, 2]. One of the principal SA features is its asymptotic convergence, but it spends a lot of computer time to find a near-optimal solution. Until nowadays, this situation has established an important investigation line where the major question is how does SA become more efficient? In a review of the state of the art [3, 4] we found that most of the works in SA area can be clustered in the following groups: Cooling schemes, Hybrid methods, Perturbation schemes and Parallel methods. The most intensive work has been done in cooling scheme and hybrid methods. That is logical because the cooling scheme establishes a good balance between efficacy and efficiency through the cooling parameters imposed. The hybrids combine SA with others combinatorial methods to ameliorate the individual algorithms. Perturbation schemes methods usually depend of particular applications. Parallelization is recognized like a powerful strategy to increase the algorithm efficiency; however SA is essentially a sequential algorithm and its parallelization becomes a hard task. Due the importance of parallelization area we developed a Methodology to Parallel the Simulated Annealing Algorithm (MPSA). Through MPSA it is possible to get an implementation of a Simulated Annealing Parallel Algorithm (SAPA) as will be defined in this paper.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 63—74, 2000.

2. Parallel Simulated Annealing Algorithms

2.1 Principal Tendencies

The works trying to do a parallel design of SA algorithm can be clustering in two approaches [for general references see 3, 4, 5, 6, 7, 8]:

- Pseudo-parallelization of SA. Sequential SA algorithms are running on different processors at the same time or the data domain is assigned to different processors where a sequential SA is running. The principal methods are: Data Configuration Partition (DCP), Parallel Independent Annealing (PIA) and Parallel Markov Chains (PMC).

- Simulated Annealing Parallel Algorithms (SAPA). SA algorithm is divided in tasks, which are distributed among several processors. Some SAPA methods are: Parallel Markov Chains (PMC), Parallel Markov Trials (PMT), Adaptive Parallel Simulated Annealing (APSA), Speculative Trees and Systolic.

Data Configuration Partition (DCP). In DCP, data are divided in *n* subsets and distributed among the processors, each of them executing a sequential SA algorithm [6, 7, 8, 9, 10, 11, 12, 13]. DCP has a big communication overhead because it requires checking the border condition compatibility among data subsets.

Parallel Independent Annealing (PIA). PIA is made running independent SA on the processors [5, 7, 9, 12, 14]. The best solution is selected as the final one. PIA has the same efficiency that the sequential SA algorithm.

Parallel Markov Chains (PMC). Several sequential SA are running on different processors but with a periodical interaction [5, 7, 8, 12, 13, 15]. PMC divides SA in Metropolis search's [16], which are modeled through Markov chains [17]. For each new Markov chain an initial solution is established as the best solution among processors. This periodical interaction lets a better performance than PIA if the communication overhead can be balanced with the Markov chain lengths. This approach has a good performance for high temperature values just.

Parallel Markov Trials (PMT). Markov trials are executed on different processors in an independent way [5, 6, 7, 8, 12, 14]. At the moment that a new solution is accepted it is transferred to a master processor which chooses one from all the accepted solutions. Here all the processors contribute to generate only one Markov Chain. This approach is a SAPA type and has a good performance only for low temperature values.

Adaptive Parallel Simulated Annealing (APSA). APSA combines PMC and PMT. PMC technique is used at high temperatures and PMT when low temperatures are gotten [5, 8]. There are also some kinds of APSA methods that adjust the number of processors in a dynamic way [18]. In general APSA methods are considered like an SAPA type because the use of PMT.

Speculative Trees. This method generates a decision tree over processors where the root has the actual solution and it can generate a proposed new solution in the branches. This tree is named speculative because the new solution can be accepted o rejected [6, 19, 20]. These kinds of algorithm considered like a SAPA method, usually has a huge communication overhead.

Systolic. This is a SAPA method where each processor is running with different temperature using the Markov chain information [6, 11, 21]. This method has some similarities with the methodology proposed here and will be explained in next section.

There are also some algorithms (SAPAs or not) that combine SA with Genetics algorithms and neuronal networks and use the intrinsic parallelism of these methods [5, 13, 22, 23, 24].

2.2 The Systolic Algorithm

Let be P the number of processors and L the length of the Markov chain. Each Markov chain is divided in P sub-chains of length $N = P/L$. The way in which the SA Algorithm is executed is as follow: the processor number one builds the first sub-chain at temperature c_1. As soon as it finishes, the processor number two still building the Markov chain at temperature c_1 through the execution of the second sub-chain. The processor one decrements the temperature to c_2 and starts the first sub-chain at this new temperature taking like actual solution the solution that it has. When the processors one and two have finished theirs sub-chain, the processor three begins its work retaking the Markov chain at c_1 temperature, then the processor two retakes its work with the Markov chain at the c_2 temperature. Then the processor one will enter decreasing the temperature below of c_2, getting c_3 and starting the Markov chain at this new temperature. This process continues until convergence.

The systolic algorithm takes worse solutions than the sequential SA algorithm [11, 21]. This occurs because each Markov chain starts before the equilibrium is gotten in the previous Markov Chain. When the number of processor is increased, the sub-chain length N decrease and the efficacy decrease even more.

3. Sequential Simulated Annealing Algorithm

The SA algorithm as [1] and [2] proposed it is implemented as follow:

```
Procedure SIMULATED ANNEALING

Begin
     INITIALIZE(S_i=initial_state, c=initial_temperature)
     k = 0
     Repeat
          Repeat
               S_j = PERTURBATION(S_i)
               If COST(S_j) ≤ COST(S_i) Then
                    S_i = S_j
```

```
                Else
                     If exp(-Δcost/c) > random[0,1) Then
                          S_i = S_j
          Until thermal equilibrium
          k = k + 1
          c = COOLING(c)
     Until stopcriterio
End
```

Analyzing the pseudo-code, we can note that two nested cycles forms the SA algorithm. The external cycle establishes a descent sequence of the control parameter (c) from the initial temperature. The internal cycle does a random walk on the solution space until the equilibrium is gotten at a fixed temperature (Metropolis algorithm) [16], i.e., a Markov chain is built until the stationary distribution is reached. The stationary distribution is established by the Boltzmann distribution. SA algorithm schematization is shown in the Fig. 1.

4. MPSA Methodology to Parallel the Simulated Annealing Algorithm

4.1 Observations

From the SA algorithm's state of the art [3, 4] we can do the next observations:

- Two nested cycles forms the SA algorithm. The external one establishes a cooling scheme and the internal one does a random walk on the solution space.
- In general, the implementation of the SA algorithm is done through homogeneous Markov chains. Individual Markov chains are built through a set of trials where an actual solution is transformed to a new one through a perturbation device and an acceptance criteria.
- The cooling scheme parameters set establishes a balance between efficacy and efficiency. Nowadays, we know cooling schemes that guarantee asymptotic convergence in a theoretical way.
- There doesn't exist a precise form to determine the Markov chain length (L). In general, L is determined in the literature through experimental techniques.
- The SA algorithm based on the physical analogy of Solid Annealing is essentially a sequential process. This fact has provoked that the design of an efficient parallel algorithm becomes a difficult task.
- The parallel SA design methods have been focused in order to make a distributed work of the internal cycle (distributed work of the Metropolis procedure).
- The systolic algorithm could be seen like a parallel design that generates Markov chain at different temperatures in a parallel way, but it takes worse solutions than the sequential SA algorithm.

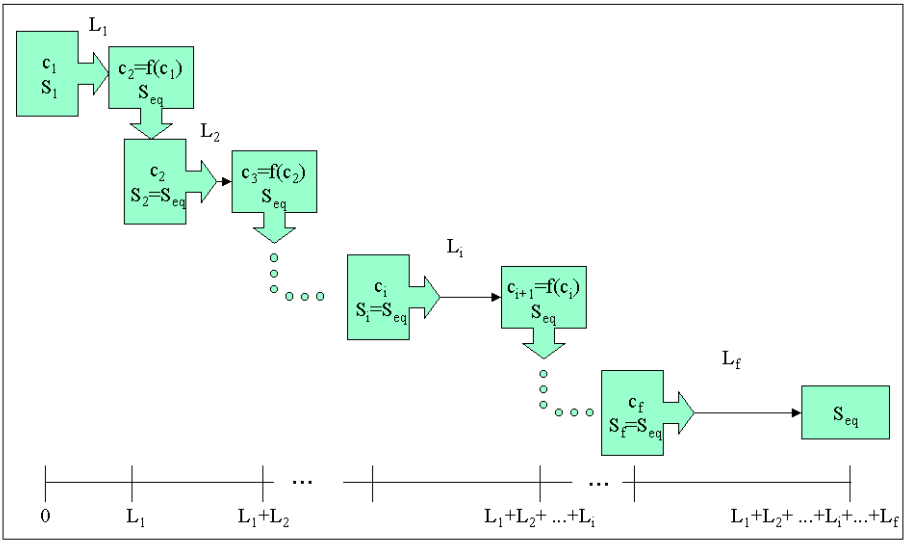


Fig. 1. Schematization of the sequential SA algorithm.

4.2 Boltzmann Distribution Function and the Simulated Annealing Algorithm

From the Statistics Mechanics we know that the distribution function in the equilibrium for a thermal system and for the SA algorithm, because the analogy between the physics system and the combinatorial optimization problem (COP), is given by the Boltzmann distribution function (BDF).

The BDF, $q(c)$, is really a family curve. Each curve is defined by the parameter c, which determines the system temperature, and establishes the states' probability density for that temperature.

When we are solving a COP. It can be shown that when the temperature is high, $c \rightarrow \infty$, we have an homogeneous distribution over all the states. But if the temperature is very low, $c \rightarrow 0$, then we have a homogeneous distribution over the set of optimal solutions. BDF behavior for different values of parameter c, taking a quadratic function like energy function is shown in Fig. 2.

Through BDF family curves we can know the probability distribution function (pdf) at the beginning and at the end of each internal cycle of the SA algorithm. The pdf at the beginning of the internal cycle at temperature c_{k+1} is given by the pdf at the end of the internal cycle for the previous temperature c_k in the descendent sequence $\{c\}$.

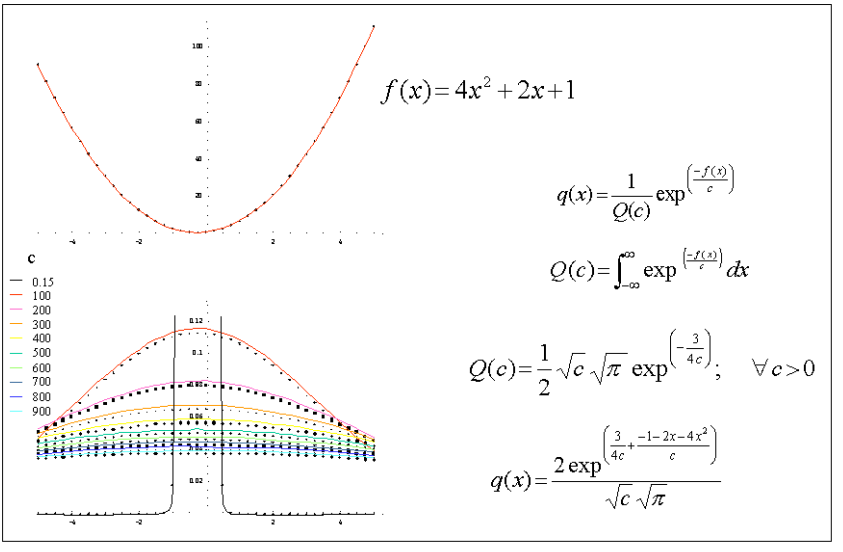


Fig. 2. Boltzmann distribution function.

A schematization of the SA algorithm through the BDF is shown in Fig. 3. It can be seen as a sequence of Markov chains in which the pdf will pass from an homogeneous distribution over all the solution space to a distribution in which the states with minimum cost value are the only ones that have an existing probability.

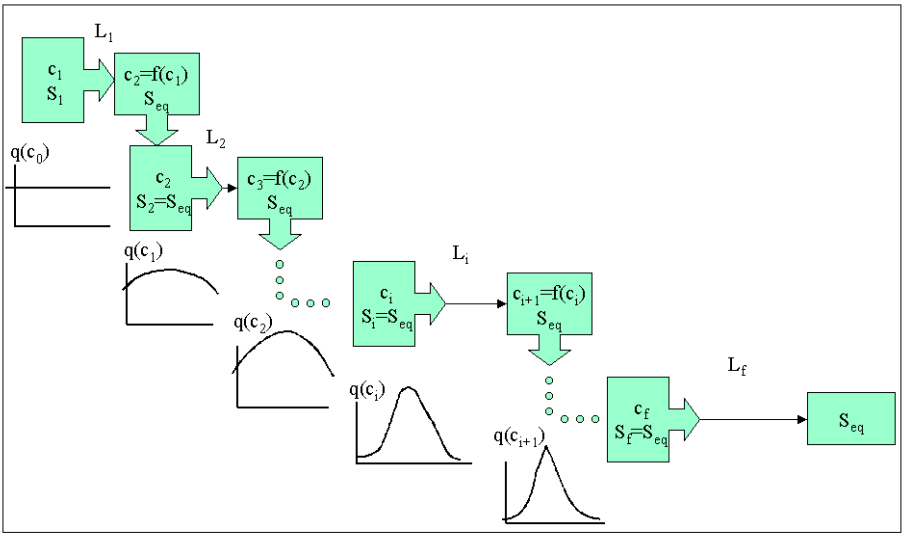


Fig. 3. Schematization of the SA algorithm through the Boltzmann distribution.

The SA algorithm schematization given at Fig. 3 shows clearly that for each internal cycle of temperature we always know the pdf, at the beginning and also at the end of it. Where the distribution function establishes the existing likely of each state into the system with respect to its cost value. This is a consequence of the Metropolis algorithm, which is given by the internal cycle of the SA algorithm. The Metropolis algorithm guarantee that the equilibrium given by the stationary distribution in a Markov process is going to be reach when the number of iterations is huge.

The pdf behavior changes with respect to the temperature (see Fig. 2) and it will be function of the problem complexity. The distribution function is smooth and has a good behavior at high temperatures. It is going to change to multi-modal function at the time that the algorithm reaches intermediate temperatures. Finally, it is going to change to a smooth function, again, until it become an impulse function (when we have just one global minimum) or an impulse train (when we have more than one global minimum) at zero temperature.

4.3 Methodology to Parallel the Simulated Annealing External Cycle

Taking into account the observation given at Sect. 4.1 and SA algorithm schematization through the BDF (see Fig. 3), a new Methodology to Parallel the Simulated Annealing Algorithm (MPSA) is developed.

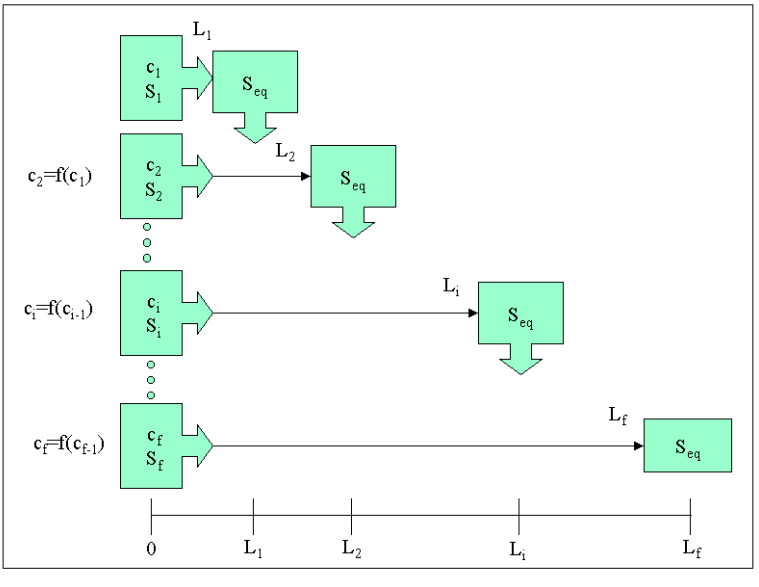


Fig. 4. Schematization for the external cycle parallelization in the SA algorithm.

The MPSA's kernel is to do an external SA algorithm cycle parallelization. The external cycle parallelization implies that several processors build their own Markov chain, each one at different temperature. That is, instead of a new Markov chain at

temperature c_{k+1} be started at the end of the Markov chain at temperature c_k all the Markov chains in the SA algorithm are started at the same time (see Fig. 4). Ideally, it lets a massive parallelization because we have a processor for each temperature in the annealing process and based in the asymptotic convergence we need an infinite temperature sequence $\{c\}$.

Any implementation that follows the steps that will be established in this MPSA methodology, will led a SAPA method, because it will build the SA algorithm's Markov chains in a parallel way.

The determination of the initial configuration for each Markov chain is a strong problem, since it must be the final solution of the Markov chain generated in the previous temperature (see Fig. 1). However, the knowledge of the initial and final distribution function for each homogeneous Markov chain can be used to infer the initial solution at each temperature.

MPSA does the inferring process for each temperature taking a Monte Carlo random sampling to adjust the pdf through a population of points. So the initial solution for each temperature is established like the best one through two criteria. The first one establishes that the solution with the highest Boltzmann probability will be taken as the initial solution, and the second criteria sets the initial solution like the configuration with minimum cost for homogeneous or quasi-homogeneous pdf (at high temperatures).

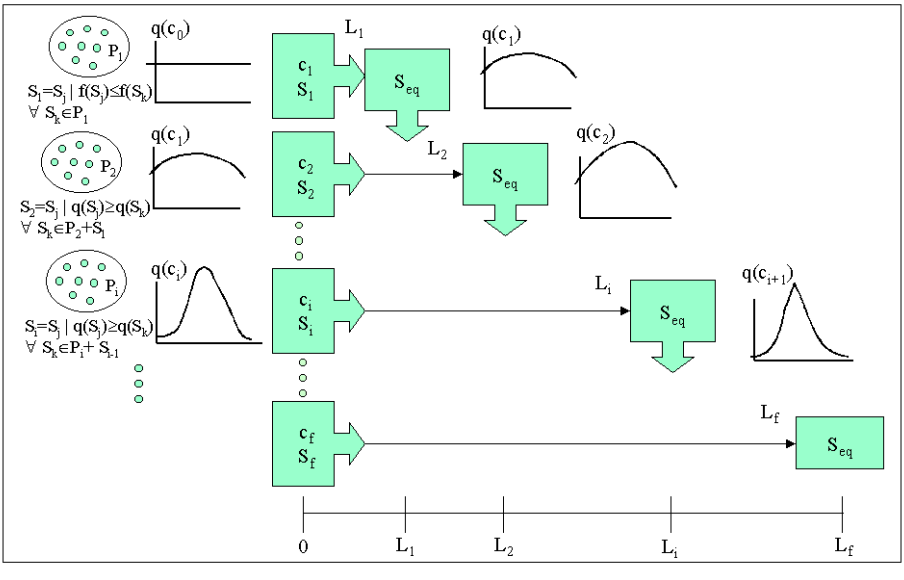


Fig. 5. Random samplings to initialize each Markov chain.

The Monte Carlo random sampling for each temperature must be taken in an independent and parallel way to reduce the sampling time. The sample size must be representative of the solution space. It must be set through the sampling theory. In this way, we have a population that adjusts the pdf at the beginning of each Markov chain. This adjustment can be better through the random samplings union generated for higher temperatures. With this unions we have a population each time bigger when the temperature is downing and the pdf is more complex (see Fig. 5). The sampling will be done off line and is part of the tuning process to set the initial solution for each temperature.

The cooling scheme parameters must be known to carry out the external cycle parallelization. They can be taken from literature works. However, we have to have care with the homogeneous Markov chain length for each internal cycle. They must be set in a way that the thermal equilibrium or quasi-equilibrium is reach at each temperature. MPSA establishes that any SAPA implementation has the same cooling scheme than the sequential SA algorithm.

Analyzing the curves in Fig. 2 and the cooling functions reported at literature, we note that the changes in the pdf form at the beginning of two consecutive Markov chains are an exponential function. As a consequence, although we will try to keep the quasi-equilibrium among different Markov chains or processors through a cooling function, the Markov chain length will grow when the temperature is descending.

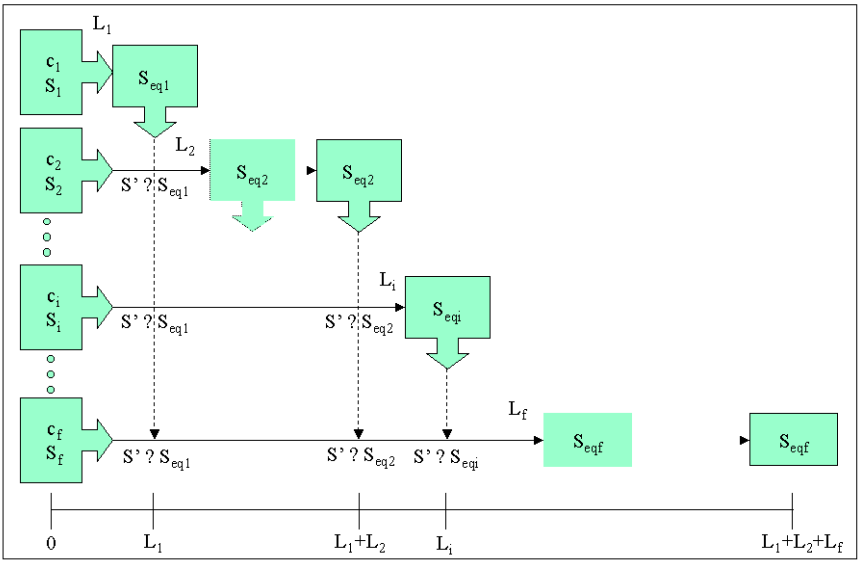


Fig. 6. Asynchronous communication scheme

The Markov chain length growing implies that any processor working at higher temperature will finish before another one. This establishes a sequence in the work

ending of the processors. MPSA use this sequence to establish the communication scheme among processors. To MPSA each processor broadcasts its final solution to all the processors that are working at lower temperature than it does. When a processor receives a solution, it checks if it is better than its actual solution; if it is the case, the processor will restart the Markov chain with this new solution, else it continues in a normal way (see Fig. 6). The first criterion to check a better solution is the biggest Boltzmann distribution probability and the second one is the cost value.

In MPSA the communications have to be asynchronous. So the processors are kept workings without spent time waiting by retro-feed; they simply take it at the moment that it arrives. An exception is when a processor has finished and any processor is even working at a higher temperature. In this case the former processor broadcasts its actual solution and wait to receive another solution from any of those processors.

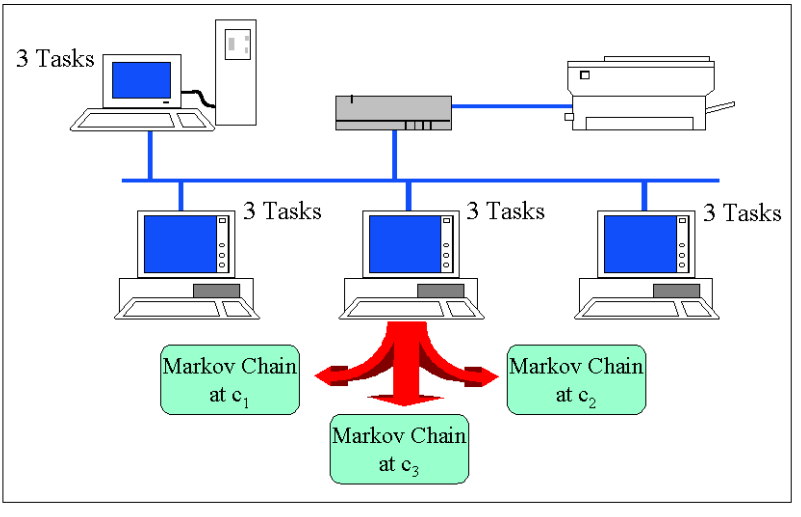


Fig. 7. Virtual parallelization into a LAN

Through this communication scheme the worst case is established when all the processors (except the first in the sequence) should re-start at the output solution of a previous one. Because we will not have a significant communication overhead the execution time of any SAPA implementation with MPSA should be close to the sequential SA algorithm version. The best case for a SAPA implementation with our methodology is when the re-starts are not required for any processor. Then, the execution time is the last cycle of temperature and a little communication overhead.

4.4 Virtual Parallelization

The MPSA parallelization needs that each processor executes a Markov chain building task that generates the homogeneous Markov chain at the temperature

assigned to the task. Like, MPSA methodology lets a massive parallelization, based in the asymptotic convergence of the algorithm, it can be implemented in a virtual way.

A virtual parallelization is established through a parallel machine or a LAN with just a few processors in which each processor execute more than one Markov chain building task. For example, Fig. 7 shows a LAN where each processor is running three tasks and each one is building a Markov chain.

The virtual parallelization reduces the total process efficiency because all the tasks running in one processor are competing by the same resources (processor time, memory, etc.). Besides, a virtual parallelization into a LAN has an overhead in the communication time because the network loads. Although this overhead may be balanced by the asynchronous communication.

5. Conclusions

We have proposed a new Methodology to Parallel Simulated Annealing (MPSA). It is based in the supposition that during the annealing process, the system reaches the thermal equilibrium at each temperature. It is based also in the fact than the parallelization of the temperature cycle ideally lets a massive parallelization of the implemented SAPA algorithm.

It has been shown that the initial solution for each internal cycle may be set through a Monte Carlo random sampling, adjusting by the Boltzmann distribution at the beginning of each cycle. MPSA imposes that sampling overhead be avoided by an off line sampling. So it is considered like a part of the tuning process of the SAPA algorithm.

The communication overhead is negligible because the communication scheme established for this methodology is asynchronous. As was shown in the paper, any SAPA implementation derived through MPSA is in general more efficient than the sequential version.

References

1. Kirkpatrik, S., Gelatt Jr., C.D. and Vecchi, M.P., 1983. Optimization by simulated annealing. Science, Vol. 220, No. 4598, pp. 671 - 220.
2. Cerny, V., 1985. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. Journal of Optimization Theory and Applications. Vol. 45, No. 1, pp. 41 - 51.
3. Sanvicente, S.H., 1997. Recocido simulado: optimización combinatoria. Estado del arte. Instituto Tecnológico y de Estudios Superiores de Monterrey (campus Morelos), México. 72 pp.
4. Sanvicente, S.H., 1998. Recocido simulado paralelo. Propuesta de Tesis Doctoral. Instituto Tecnológico y de Estudios Superiores de Monterrey (Campus Morelos), México, 38 pp.
5. Aarts, E. and Korst, J., 1989. Simulated annealing and Boltzmann machines: A stochastic approach to combinatorial optimization and neural computing. John Wiley & Sons, Great Britain, 272pp.

6. Greening, D.R., 1990. Parallel simulated annealing techniques. Physica D., Vol. 2, pp. 293 – 306.
7. Azencott, R., 1992. Simulated Annealing: Parallelization techniques. R. Azencott (Editor). John Wiley & Son, USA, 200pp.
8. Diekmann, R., Lüling, R. and Simon, J., 1993. Problem independent distributed simulated annealing and its applications. Tech. Report No. TR-003-93. Department of Mathematics and computer Science, University of Paderborn, Germany, 23 pp.
9. Krishna, K., Ganeshan, K. and Ram, D.J. 1995. Distributed simulated annealing algorithms for job shop scheduling. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 25, No. 7, pp. 1102 – 1109.
10. Voogd, J.M. and Sloot, P.M.A., 1993. Crystallization on a sphere using the simulated annealing algorithm implemented on H.P.C. systems. Technical Report: PSCSG-93-01, Parallel Scientific Computing and Simulation Group, University of Amsterdam, 6 pp.
11. Voogd, J.M., Sloot, P.M.A. and Dantzing, R.V., 1994. Simulated annealing for N-body systems. Technical Report: PSCSG-94-01, Parallel Scientific Computing and Simulation Group, University of Amsterdam, 6 pp.
12. Dowsland, K.A., 1993. Simulated annealing. In: C.R. Reeves (Editor): Modern heuristic techniques for combinatorial problems. John Wiley and Sons, Great Britain, pp. 20 – 69.
13. Ingber, L., 1993. Simulated Annealing: Practice versus theory. J. Mathl. Comput. Modelling, Vol. 18, No. 11, pp. 29 – 57.
14. Roussel-Ragot, P. and Dreyfus, G., 1992. Parallel annealing by multiple trials: an Experimental study on a transputer network. In: R. Azencott (Editor): Simulated annealing: parallelization techniques. John Wiley & sons, USA, pp. 91 – 108.
15. Graffigne, C., 1992. Parallel annealing by periodically interacting multiple searches: An experimental study. In: R. Azencott (Editor): Simulated annealing: Parallelization techniques. John Wiley & Sons, USA, pp. 47 – 79.
16. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E., 1953. Equation of state calculations by fast computing machines, Journal of Chemical Physics, Vol. 21, pp. 1087 - 1092.
17. Kemeny, J.G. and Snell, J.L., 1965. Finite Markov chains. D. Van Nostrand Company, Inc., USA. 210 pp.
18. Virot, B., 1992. Parallel annealing by multiple trials: Experimental study of a chip placement problem using a sequent machine. In: R. Azencott (Editor): Simulated annealing: Parallelization techniques. John Wiley & Sons, USA, pp. 109 – 127.
19. Nabhan, T.M. and Zomaya, A.Y., 1995. A parallel simulated annealing algorithm with low communication overhead. IEEE Transactios on Parallel and distributed Systems, Vol. 6, No. 12, pp. 1226- 1233.
20. Sohn, A. 1995. Parallel N-ary speculative computation of simulated annealing. IEEE Transactions on Parallel and Distributed systems, Vol. 6, No. 10, pp 997 – 1005.
21. Voogd, J.M., Sloot, P.M.A. and Dantzing, R.V., 1995. Comparison of vector and parallel implementations of the simulated annealing algorithm. Technical Report: PSCSG-95-01, Parallel Scientific Computing and Simulation Group, University of Amsterdam, 11 pp.
22. Marroqin, J. L. Botello, S. and Horebeek, J., 1995. A family of parallel search algorithms. Memorias de la 12va. Reunión Nacional de Inteligencia artificial de la SMIA, pp. 164 – 171.
23. Chen, H., Flann, N.S. and Watson, D.W., 1998. Parallel genetic simulated annealing: a massively parallel SIMD algorithm. IEEE Transactions on Parallel and Distributed Systems, Vol. 9, No. 2, pp. 126 – 136.
24. Mutalik, P.P., Knight, L.R., Blaton, J.L. and Wainwright, R.L., 1992. Solving combinatorial optimization problems using parallel simulated annealing and parallel genetic algorithms. ACM 0-89791-502-X/92/0002/1031, pp. 1031 – 1038.

Vertical Fragmentation and Allocation in Distributed Databases with Site Capacity Restrictions Using the Threshold Accepting Algorithm[1]

Joaquín Pérez[1,2], Rodolfo Pazos[2], Juan Frausto[3], David Romero[4], and Laura Cruz[5]

[1] Instituto de Investigaciones Eléctricas
jperez@iie.org.mx
[2] Centro Nacional de Investigación y Desarrollo Tecnológico
pazos@cenidet.edu.mx
[3] ITESM Campus Morelos
jfrausto@campus.mor.itesm.mx
[4] Instituto de Matemáticas, UNAM
david@matcuer.unam.mx
[5] Instituto Tecnológico de Cd. Madero
lcruz@itcm.edu.mx

Abstract. This paper presents an extension of the DFAR mathematical optimization model, which unifies the fragmentation, allocation and dynamical migration of data in distributed database systems. The extension consists of the addition of a constraint that models the storage capacity of network sites. This aspect is particularly important in large databases, which exceed the capacity of one or more sites. The Threshold Accepting Algorithm is a variation of the heuristic method known as Simulated Annealing, and it is used for solving the DFAR model. The paper includes experimental results obtained for large test cases.

Keywords: Heuristic optimization, distributed database design.

1 Introduction

The generalized use of computer communication networks, including Internet, facilitates the implementation of distributed database systems (DDBS). However, the lack of methodologies and tools to support their design [1] hinders their development.

Traditionally, the distribution design (DD) has been considered to consist of two separate phases: fragmentation and fragment allocation [2]. An innovative model that integrates both phases is presented in [3]. In this paper the model is extended to deal with complex conditions that limit the normal operation of distributed database systems.

[1] This research was supported in part by the CONACYT grant No. 26351A and the COSNET grant No. 568.97-P.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 75—81, 2000.

2 Mathematical Model

In this section, a (binary) integer-programming model for the dynamic fragmentation, allocation and reallocation of data is described. We refer to this model as the DFAR model. In this model, the decision about storing an attribute m in site j is represented by a binary variable x_{mj}. Thus $x_{mj} = 1$ if m is stored in j, and $x_{mj} = 0$ otherwise.

2.1 Objective Function

The objective function below models the data transmission and access costs using four terms: transmission, access to fragments, fragment storage, and fragment migration.

$$\min z = \sum_k \sum_i f_{ki} \sum_m \sum_j q_{km} l_{km} c_{ij} x_{mj} + \sum_i \sum_k \sum_j c_1 f_{ki} y_{kj} + \sum_j c_2 w_j + \sum_m \sum_i \sum_j a_{mi} c_{ij} d_m x_{mj} \quad (1)$$

where

f_{ki} = emission frequency of query k from site i;

q_{km} = usage parameter, $q_{km} = 1$ if query k uses attribute m, otherwise $q_{km} = 0$;

l_{km} = number of communication packets needed to transport attribute m required by query k

= $(p_m s_k)/PA$

where

p_m = size in bytes of attribute m;

s_k = selectivity of query k (number of tuples returned when query k is executed);

PA = communication packet size in bytes;

c_{ij} = communication cost between site i and site j;

c_1 = cost for accessing several fragments to satisfy a query; e.g., the processing cost of a join in a query that accesses two or more fragments;

y_{kj} = decision variable, $y_{ki} = 1$ if query k accesses one or more attributes located at site j, and $y_{kj} = 0$ otherwise;

c_2 = cost for allocating a fragment to a site; e.g., the cost incurred for having a copy of the key in each fragment;

w_j = decision variable, $w_i = 1$ if there exist one or more attributes at site j, and $w_i = 0$ otherwise;

a_{mi} = indicator of previous allocation, $a_{mi} = 1$ if attribute m is currently located at site i, otherwise $a_{mi} = 0$;

d_m = number of communication packets required to move attribute m to another site if necessary;

= $(p_m * CA)/PA$

where

CA = cardinality of the relation.

2.2 Intrinsic Constraints of the Problem

Since replication of attributes is not considered, these constraints specify that each attribute is going to be allocated to only one site. Additionally, each attribute must be allocated to a site that executes at least one query involving the attribute. These constraints are as follows:

1) $\sum_j x_{mj} = 1$ $\forall m$ — each attribute must be stored in only one site;

2) $x_{mi} \leq \sum_k q_{km}\varphi_{ki}$ $\forall m, i$ — where $\varphi_{kt} = \begin{cases} 1, & \text{if } f_{ki} > 0 \\ 0, & \text{if } f_{ki} = 0 \end{cases}$ each attribute m must be allocated to a site i that executes at least one query involving the attribute;

3) $t\ w_j - \sum_m x_{mj} \geq 0$ $\forall j$ — this constraint forces the value of w_j to 1 when any x_{mj} equals 1, and induces w_j to 0 otherwise; where t = number of attributes;

4) $t\ y_{kj} - \sum_m q_{km} x_{mj} \geq 0$ $\forall k, j$ — this constraint forces y_{ki} to 1 when any $q_{km}x_{mi}$ equals 1, and induces y_{kj} to 0 otherwise.

5) $\sum_m x_{mj}\, p_m CA \leq cs_j$ $\forall j$ — the space occupied by all attributes stored in site j must not exceed the site capacity; where cs_j = capacity of site j.

3 Solution Method

This section presents the implemented algorithm, and the experimental results. Since the DFAR problem is *NP*-hard [4], a heuristic method is explored.

3.1 Threshold Accepting Algorithm

A variation of the simulated annealing algorithm (known as Threshold Accepting) was implemented. This is a simplified version of simulated annealing, and consumes less computing time than the original simulation annealing method [5].

In this algorithm, to each $\boldsymbol{x} \in \boldsymbol{X}$ (where $\boldsymbol{X}$ represents the set of all feasible solutions) we associate a neighborhood $H(\boldsymbol{x}) \subset \boldsymbol{X}$. Thus, given a current solution $\boldsymbol{x}$, a neighboring solution $\boldsymbol{y} \in H(\boldsymbol{x})$ is generated; if $\Delta z = z(\boldsymbol{y}) - z(\boldsymbol{x}) < T$ (where T is a properly chosen parameter), then $\boldsymbol{y}$ is accepted becoming the new current solution, otherwise the current solution remains unchanged.

The value of T is decreased each time a situation of *thermal equilibrium* is reached. The value of T is reduced, by repeatedly multiplying it by a *cooling factor* $\mu < 1$ until the system is frozen.

Algorithm

```
Begin
  real T, μ
  integer i, S
  x = best of a set of S feasible solutions
  T = high initial value of temperature
  repeat
    repeat
      for i = 1 to S
        y = neighboring solution
        if z(y) - z(x) < T then
          x = y
        else
          the new solution is not accepted
        end_if
          i = i + 1
      end_for
    until thermal equilibrium is reached
    T = μT
  until freezing is reached
End
```

3.2 Experimental Results

A set of exploratory experiments were conducted, in order to determine the best strategy to generate large test cases, feasible solutions, and the algorithm regulation parameters.

Large Test Cases. Large size problems were generated using the following mathematical trick: each new problem is generated combining several small problems with known optimal solution, in such a way that the restrictions of each problem are uncoupled from those of the other problems. Therefore, the optimal solution of the new problem can be easily obtained adding the optimal solutions of its sub-problems.

Feasible Solutions. When using the Threshold Accepting Algorithm on the DFAR model, sometimes the capacity limitation of sites prevents attributes to interchange positions when looking for a neighboring feasible solution. This difficulty is overcome modifying the original problem as follows:
a) **Artificial site**. An artificial site h is included with enough capacity to temporarily accommodate the attributes that can not be stored elsewhere.

b) **Objective function**. A penalization term is included in the objective function. This term penalizes those solutions in which the artificial site stores one or more attributes. Therefore the new objective function becomes

$$\begin{aligned} \min z = & \sum_k \sum_i f_{ki} \sum_m \sum_j q_{km} l_{km} c_{ij} x_{mj} + \sum_i \sum_k \sum_j c_1 f_{ki} y_{kj} \\ & + \sum_j c_2 w_j + \sum_m \sum_i \sum_j a_{mi} c_{ij} d_m x_{mj} + p(X,Y,W) \end{aligned} \tag{2}$$

c) **Penalization function**. An adequate penalization function permits the transition from unfeasible to feasible solutions [6,7]. The penalty is determined relaxing the objective function for the attributes stored in the artificial site *h*. To this end, the maximum values of the transmission *CT*, migration *CM* and access cost *CJ* coefficients are used.

$$\begin{aligned} p(X,Y,W) = & \sum_m max\{CT_{mj}, \forall j\} x_{mh} + \\ & \sum_k max\{CJ_{kj}, \forall j\} \, min(\sum_m x_{mh}, \sum_{j \neq h} (1 - y_{kj})) + \\ & c_2 min(\sum_m x_{mh}, \sum_{j \neq h} (1 - w_j)) + \sum_m max\{CM_{mj}, \forall j\} x_{mh} \end{aligned} \tag{3}$$

Algorithm Parameters. Except for the initial temperature *T*, the other problem parameters were obtained using the best strategies described in [3]. A better method was found for determining *T*: first a set of 100 neighboring feasible solutions is generated, then the objective function differences of each pair of consecutive solutions are stored in a list, and finally *T* is assigned the sixth largest value in the list. When reducing the initial temperature, smaller processing times are achieved.

Table 1. Exact Solution

Problem	Sites	Optimal Value	Time (sec.)
P1	2	302	0.05
P2	18	2719	1.15
P3	20	3022	3.29
P4	64	9670	*
P5	128	19340	*
P6	256	38681	*
P7	512	77363	*

*Problem size exceeds Lindo capacity

Test Results. The algorithm for the DFAR model was written in Borland C, and it was run on a PC with a 120 MHz Pentium processor and Windows´95 operating system.

The second column of Table 1 shows problem sizes for the test cases. The last two columns show the results obtained using the Lindo package, which is based on a mathematical programming method known as Branch&Bound method.

Table 2 presents the results obtained using the Threshold Accepting Algorithm. Forty runs were made for each problem. The percentage difference with respect to the optimal solution is shown on columns 2 through 4. The last column shows the execution time of the algorithm.

Table 2. Approximate Solution

Problem	% Difference			Time
	Best	Worst	Average	(sec.)
P1	0	0	0	0.03
P2	0	141	10	0.3
P3	0	0	0	0.4
P4	0	100	20	6.1
P5	0	140	36	43.6
P6	0	405	88	381.2
P7	66	383	215	3063.4

4 Final Remarks

This paper shows the possibility of expanding the DFAR model for incorporating site capacity constraints. This aspect is particularly important in large databases, which must be distributed among several sites as a result of their size.

The initial experiments revealed that when site capacity is used almost to its limit, attribute deadlocks might occur (i.e., interchange of attribute positions might be blocked), preventing the Threshold Accepting Algorithm to converge. The paper shows that the inclusion of an artificial site and the addition of a penalization term to the objective function, prevents attribute deadlocks and permits the algorithm convergence.

The proposed mechanism for the generation of large test cases, produced larger test cases than those reported by other authors, and allowed us to compare the results obtained using the Threshold Accepting Algorithm with the optimal results of the test problems. A contribution of this work is to provide a framework for evaluating approximate solution methods for the distribution design problem; specially, for heuristic methods for which there exists no frameworks.

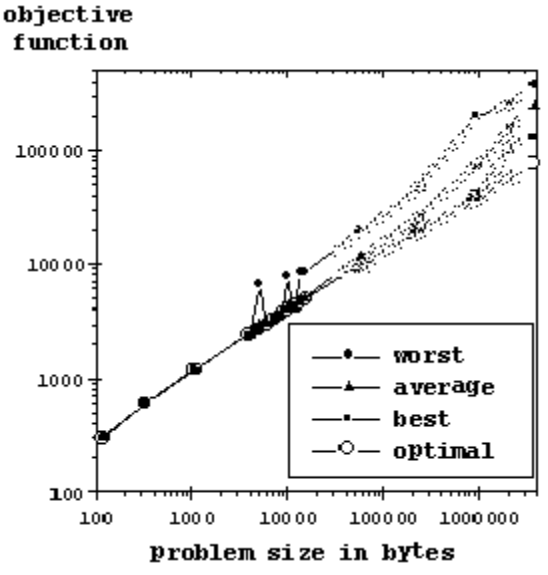


Fig. 1. Results Using the Threshold Accepting Algorithm

This paper shows the feasibility of using the Threshold Accepting algorithm, which showed good performance for small problems and demonstrated its ability to solve large problems. However, the solution tends to grow farther as the problem size increases (Table 2 and Fig. 1).

References

1. C. García Hong-Mei, O. R. Liu Sheng: A Semantic Based Methodology for Integrated Computer-Aided Distributed Database Design. In: Proc. 25th Hawaii International Conference on System Sciences, Vol. 3, (1992) 288-299.
2. S. Ceri, G. Pelagatti: Distributed Databases: Principles & Systems. McGraw-Hill, New York. N.Y. (1984).
3. J. Pérez, D. Romero, J. Frausto, R. Pazos, G. Rodríguez, F. Reyes: Dynamic Allocation of Vertical Fragments in Distributed Databases Using the Threshold Accepting Algorithm. In: Proceedings of the 10th IASTED International Conference on Parallel and Distributed Computing and Systems, Las Vegas (1998) 210-213.
4. X. Lin, M. Orlowska, Y. Zhan: On Data Allocation with the Minimum Overall Communication Cost in Distributed Database Design. In: Proc. of ICCI'93 (1988) 539-544.
5. L. Morales, R. Garduño, D. Romero: "The Multiple-Minima Problem in Small Peptides Revisited, The threshold Accepting Approach. In: Journal of Biomolecular Structure & Dynamics, Vol. 9, No. 5 (1992) 951-957.
6. R. Pazos: Evaluation and Design of Integrated Packet Switching and Circuit Switching Computer Networks. Ph.D. dissertation, Computer Science Dept., UCLA (december 1983) 236.
7. J. F. Beasley, K. Dowsland, F. Glover, M. Laguna: Modern Heuristic Techniques for Combinatorial Problems. New York: Colin R. Reeves (1993) 320.

A Practical Approach for Logic Program Analysis and Transformation

Wamberto Weber-Vasconcelos and Eudenia Xavier Meneses

Departmento de Estatística e Computação
Universidade Estadual do Ceará
Av. Paranjana, 1700 – 60.740-000 Fortaleza Ceará, Brazil
wvasconcelos@acm.org, eudenia@uece.br

Abstract. In this work we define a schematic language to concisely describe classes of logic programs. We show how our proposed formalism can be profitably employed to represent *opportunities* for program optimisation: we schematically specify commonly occurring inefficient portions of code and how these can be altered to improve the performance of programs. We have defined a fully automated approach to exploit these opportunities for the analysis and transformation of logic programs.

Keywords. Logic program analysis & transformation, program schemata.

1 Introduction

Research on program analysis [6] and transformation [7] aims at providing software practitioners with tools and methods for the development of correct and efficient computer systems. Within logic programming a wealth of theoretical results [8] have been accumulated over the years but they have not had the practical impact one would have wished. The work described here remedies this: we propose a practical approach for logic program analysis and transformation providing fully automated support to handle actual programs.

Given a logic program Π we want to find an equivalent program Π' (that is, which provides the same solutions as Π) but whose computational performance is better. A program outperforms another when it uses less memory and/or computes solutions in less time. For instance, let us consider the following Prolog program *sum*/2

```
sum([],0).
sum([H|T],S):- sum(T,ST), S is ST + H.
```

It correctly computes in its second argument the sum of the elements of the list in the first argument. However, its computational behaviour has a bad feature: the program has a non-tail-recursive clause, that is, its second clause has a recursive subgoal which is not the last one. The execution of non-tail-recursive clauses in Prolog requires that a stack be built to record each instance of the recursive call [1] – memory and time will be consumed in this process. Moreover, for big instances of lists, *sum*/2 above will be aborted with a stack overflow error. We can solve this problem if we appropriately insert an accumulator pair [10], thus obtaining

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 82–94, 2000.

```
sum(L,S):- sum(L,0,S).
sum([],S,S).
sum([H|T],Acc,S):- NAcc is Acc + H, sum(T,NAcc,S).
```

In this new version, we will have the same results as the previous program, *i.e.*, its declarative meaning is preserved, but the results are computed differently, *i.e.*, its procedural meaning is distinct. Since the only recursive clause (*i.e.*, the third one) is now tail-recursive, no stack is needed – its execution is just like an ordinary loop of a procedural language and is hence faster than that of the original program. Furthermore, any instances of lists can be processed without errors.

The opportunity for improvements shown above can be represented in a generic way, by means of a *transformation* where the inefficient constructs are described as well as the changes that have to be made in order to confer a better performance on the programs in which they appear. This idea was originally presented in [5], for functional programming, and in [4] we see the same idea applied to logic programs. In [13], a completely automated approach is proposed for the analysis and transformation of logic programs incorporating this concept, and in [11], a more efficient method is proposed to perform the analysis of programs. In this work we employ a schema language to describe classes of logic programs, being as generic as possible but without losing precision. Our schema language also permits the representation of *constraints* to express non-syntactic properties. Our schema language is based on the language originally proposed in [13] and expanded in [3].

In the next Section we introduce the schema language. In Subsection 2.1 we describe the process of *semi-unification* in which a match between a program schema and an actual program takes place. In Section 3 we show how our language is made more expressive by the representation of non-syntactic properties of schemata via constraints. In Section 4 we show how our schema language with constraints can be used to define opportunities for logic program transformations. In Section 5 we explain a bottom-up method for program analysis and transformation which exhaustively tries to match opportunities against actual programs. Finally in Section 6 we present our conclusions and directions for future work.

2 The Program Schema Language

We would like to describe classes of logic programs suppressing their unimportant features but retaining precision. In order to achieve this we propose a schematic language in which an abstract notation is offered to generically represent programming constructs. Some abstract symbols and their intended meaning are presented in Table 1. All sequence symbols of the language can represent, in particular, empty sequences. The predicate and function variables are meta-variables generically representing, respectively, predicate symbols and functors[1].

We define a logic program schema $\mathcal{S}$ as a sequence of schematic clauses and vectors $\boldsymbol{C}_i$ of clauses. Schematic clauses, if present in a schema, are of the

[1] Predicate and function symbols will be identified in an unambiguous manner by their symbols, their arities being implicit in their names.

Table 1. Schematic Symbols and Their Intended Meaning

Symbol	Meaning
$\boldsymbol{C}_i$	finite sequence (*vector*) of clauses
$\boldsymbol{G}_i$	finite sequence (*vector*) of literals (subgoals)
$\boldsymbol{A}_i$	finite sequence (*vector*) of terms
P, Q, R, S, T	predicate variables
F, G, H	function variables
t	terms

form $H \leftarrow B_1, \ldots, B_n$, where H and $B_i, 1 \leq i \leq n$, are schematic literals, and the B_i's can also be vectors $\boldsymbol{G}_j$ of literals. A schematic literal is of the form $p(SchTrms)$, where p is a specific predicate symbol, or of form $P(SchTrms)$, where P is a predicate variable. *SchTrms* is a possibly empty sequence of terms (*i.e.*, variables, constants or complex terms), vectors of terms $\boldsymbol{A}_i$, or constructs $F(SchTrms)$ where F is a functional variable. Actual logic programming constructs are also allowed in our schema language: specific constructions restrict the class of programs represented. In Appendix A we offer a formal definition for our schema language. We show in Figure 1 an illustrative example of a logic program schema (we enclose schemata in boxes to improve their visualisation). The schema language allows the economic representation of classes of logic pro-

$$\boxed{\begin{array}{l} P(\boldsymbol{A}_1, t, \boldsymbol{A}_2) \leftarrow \boldsymbol{G}_1 \\ P(\boldsymbol{A}_3, F(\boldsymbol{A}_4, y, \boldsymbol{A}_5), \boldsymbol{A}_6) \leftarrow \boldsymbol{G}_2, P(\boldsymbol{A}_7, y, \boldsymbol{A}_8), \boldsymbol{G}_3 \end{array}}$$

Fig. 1. Sample Logic Program Schema

grams. If we associate appropriate values with each schematic symbol, we can obtain different actual programs.

2.1 Abstraction, Specialisation, and Semi-unification

A program schema represents a possibly infinite set of programs, each of which obtained by specifying schematic constructs. On the other hand, given a program we can obtain a possibly infinite number of schemata to represent it, with different degrees of abstraction of its constructs. We formally represent the relationships between schemata and programs through operator $\delta : \mathcal{S} \times \Pi \times \Theta$, where $\mathcal{S}$ is a schema, Π is a program and Θ is a *semi-unification*. A semi-unification is a set of pairs *SchSymb*/*ProgConstr*, where *SchSymb* is a schematic symbol and *ProgConstr* is an actual logic program construct. The application of a semi-unification Θ to a schema $\mathcal{S}$, denoted by $\mathcal{S}\Theta$, consists of, given a pair *SchSymb*/*ProgConstr* in Θ, uniformly replacing in $\mathcal{S}$ all occurrences of *SchSymb* by *ProgConstr*, obtaining program Π. The relation $\delta(\mathcal{S}, \Pi, \Theta)$ holds if, and only if, $\Pi = \mathcal{S}\Theta$.

The association between the pairs of a semi-unification is subject to the meaning (semantic) of each schematic construct. The semantic restricts the pos-

sibilities of matching. For example, we can only associate a vector of clauses $\boldsymbol{C}_i$ with a possibly empty sequence of program clauses; a vector of literals $\boldsymbol{G}_i$ can only be associated with a possibly empty sequence of actual program subgoals. We show in Appendix A a formal definition for δ. To illustrate this point we show below a schema, a program and a semi-unification which satisfy the δ relationship. Sequences of constructs are shown enclosed in "$\langle$" and "$\rangle$", the empty sequence represented as "$\langle\ \rangle$":

$$\delta\left(\boxed{\begin{array}{l} P(\boldsymbol{A}_1, t, \boldsymbol{A}_2) \leftarrow \boldsymbol{G}_1 \\ P(\boldsymbol{A}_3, F(\boldsymbol{A}_4, y, \boldsymbol{A}_5), \boldsymbol{A}_6) \leftarrow \\ \quad \boldsymbol{G}_2, P(\boldsymbol{A}_7, y, \boldsymbol{A}_8), \boldsymbol{G}_3 \end{array}}, \boxed{\begin{array}{l} \texttt{sum([],0)} \leftarrow \\ \texttt{sum([H|T],S)} \leftarrow \\ \quad \texttt{sum(T,ST),} \\ \quad \texttt{S is ST + H} \end{array}}, \left\{\begin{array}{c} P/\texttt{sum}^2, \boldsymbol{A}_1/\langle\ \rangle, t/\texttt{[]}, \boldsymbol{A}_2/\langle\texttt{0}\rangle, \boldsymbol{A}_3/\langle\ \rangle, y/\texttt{T} \\ F/., \boldsymbol{A}_4/\langle\texttt{H}\rangle, \boldsymbol{A}_5/\langle\ \rangle, \boldsymbol{A}_6/\langle\texttt{S}\rangle, \boldsymbol{A}_7/\langle\ \rangle, \\ \boldsymbol{A}_8/\langle\texttt{ST}\rangle, \boldsymbol{G}_1/\langle\ \rangle, \boldsymbol{G}_2/\langle\ \rangle, \boldsymbol{G}_3/\langle\texttt{S is ST + H}\rangle, \end{array}\right\}\right)$$

If we replace the schematic constructs by the respective program components as stated in the semi-unification, we will obtain the program (specialisation). If, on the other hand, we replace the program components by the respective schematic constructs as stated in the semi-unification we shall get the schema (abstraction).

We relate a schema with a program via a *semi*-unification because we do not assign terms to first-order variables as is the case in unification [2]. The δ operator can only associate program variables with schema first-order variables. However, in the schema of Figure 1, we can see the occurrence of a *term* t in the head of the first clause: in such a case we can associate with t any logic program term, *i.e.*, variables, constants or compound terms.

Given a schema $\mathcal{S}$, the δ operator relates it with a possibly infinite set of programs $\{\Pi_1, \ldots\}$, each of which has its own semi-unification $\{\Theta_1, \ldots\}$ such that $\Pi_i = \mathcal{S}\Theta_i$. Given a program Π, the δ operator relates it with a possibly infinite set of schemata $\{\mathcal{S}_1, \ldots\}$, and a set of semi-unifications $\{\Theta_1, \ldots\}$, such that $\Pi = \mathcal{S}_i\Theta_i$. However, given $\mathcal{S}$ and Π, the δ operator relates them with a necessarily finite set of semi-unifications $\{\Theta_1, \ldots, \Theta_n\}$, such that $\Pi = \mathcal{S}\Theta_i$. In such a case we have a finite set of semi-unifications because both the program and the schema have finite sequences of constructs and, since a semi-unification simply makes a correspondence between schematic components (necessarily finite) and program constructs (also finite), the set of all possible forms of relating them is finite.

The complexity involved in obtaining a semi-unification is a function of the kinds of abstract constructs offered in the schema language. Since our schema language does not allow the representation of permutations of program constructs, the complexity of the semi-unification process may be described as a polynomial function on the number of schematic and program constructs. However, if we allow the representation of permutations of program constructs in the schema language as in the proposal of [3], the computational complexity of the semi-unification will become exponential.

3 Adding Constraints to the Schema Language

Any of the actual constructions of logic programs can be part of a schema and since these constructions are specific they can be understood as restrictions to semi-unifications. However, in [13] the authors observe that certain additional

non-syntactic constraints may be necessary to further restrict the class of programs represented by a schema. If, for instance, we wanted to restrict the class of programs represented by the schema of Figure 1 so as not to include programs with doubly recursive clauses, then we could add extra *constraints* to the schema, posing restrictions on its possible semi-unifications. If we added to the schema of Figure 1 the non-syntactic constraints $\langle P(\boldsymbol{A}_9)\rangle \not\sqsubseteq \boldsymbol{G}_1 \wedge \langle P(\boldsymbol{A}_{10})\rangle \not\sqsubseteq \boldsymbol{G}_2 \wedge \langle P(\boldsymbol{A}_{11})\rangle \not\sqsubseteq \boldsymbol{G}_3$ stating that the sequences of program literals semi-unified with $\boldsymbol{G}_1$, $\boldsymbol{G}_2$ and $\boldsymbol{G}_3$ should not have any occurrence of $P(\boldsymbol{A}_i)$ then we rule out the possibility that the first schematic clause matches a recursive program clause and that the second schematic clause matches a program clause that had more than one recursive call. Our new schema, added with these constraints, will be that of Figure 2. We can associate a possibly empty set of constraints with each

$P(\boldsymbol{A}_1, t, \boldsymbol{A}_2) \leftarrow \boldsymbol{G}_1$ $P(\boldsymbol{A}_3, F(\boldsymbol{A}_4, y, \boldsymbol{A}_5), \boldsymbol{A}_6) \leftarrow \boldsymbol{G}_2, P(\boldsymbol{A}_7, y, \boldsymbol{A}_8), \boldsymbol{G}_3$
$\langle P(\boldsymbol{A}_9)\rangle \not\sqsubseteq \boldsymbol{G}_1 \wedge \langle P(\boldsymbol{A}_{10})\rangle \not\sqsubseteq \boldsymbol{G}_2 \wedge \langle P(\boldsymbol{A}_{11})\rangle \not\sqsubseteq \boldsymbol{G}_3$

Fig. 2. Logic Program Schema with Constraints

schema and each constraint describes a desired property of semi-unification pairs: only those semi-unifications satisfying the constraints are considered correct.

In [3], we find a proposal for such constraints. Our language is based on this proposal as well as on the original idea found in [13]. Our constraints are:

- *Size Constraints* – of the form $|SchVec| \circ n$, where $SchVec$ is any vector $\boldsymbol{C}$, $\boldsymbol{G}$, or $\boldsymbol{A}$, $\circ$ is $=, <, >, \leq$ or $\geq$ and n is an integer or a variable. Such constraints are satisfied if the size of $SchVec$ satisfies relation $\circ$ with n.
- *Contents Constraints* – of the form $\langle SchComp_1, \ldots, SchComp_n\rangle \sqsubseteq SchVec$, where $SchComp_i$, $1 \leq i \leq n$, are schematic components, *i.e.*, clauses, goals, terms, and $SchVec$ is as above. Such constraints are satisfied if the subsequence $\langle SchComp_1, \ldots, SchComp_n\rangle$ occurs in $SchVec$.
- *Difference Constraints* – of the form $SchProg_1 = SchProg_2 - \{C_1, \ldots, C_n\}$, where $SchProg_1$ and $SchProg_2$ are program schemata and $C_i, 1 \leq i \leq n$, are schematic clauses or vectors of clauses. This constraint is verified if $SchProg_1$ is equal to $SchProg_2$ after $\{C_1, \ldots, C_n\}$ are removed.
- *Union Constraints* – of form $SchProg_1 = SchProg_2 \sqcup SchProg_3$, where $SchProg_i$, $1 \leq i \leq 3$, are as above. This constraint is satisfied if $SchProg_1$ is equal to $SchProg_2$ added with $SchProg_3$.
- *Definition Constraints* – of form $P =_{def} SchProg$, where P is a predicate variable and $SchProg$ is a program schema. This constraint is verified if the predicate definition for P, *i.e.*, all clauses whose head goal has predicate symbol P match $SchProg$.
- *Replacement Constraints* – of form $SchVec_1 = SchVec_2 \bullet \{x_1/Term_1, \ldots, x_n/Term_n\}$. This constraint is satisfied if $SchVec_1$ is equal to $SchVec_2$ after all occurrences of variable x_i are consistently replaced by $Term_i$.

Negated forms ($\not\sqsubseteq \neq$) for our constraints are also offered. Although the choices of constraints added to our schema language are not as numerous as those offered

in the language of [3], they can be combined into richer expressions. We want our added constraints to restrict the possibilities of semi-unification between a program and a schema and we shall denote this by the expression $\lceil \mathcal{S}\Theta = \Pi \rceil^{\mathcal{R}}$ stating that schema $\mathcal{S}$ semi-unifies with program Π through semi-unification Θ subject to constraints $\mathcal{R}$. In Appendix A.1 we formally define the meaning of our constraints.

4 Schema Language with Constraints for Program Transformation

Our schema language with constraints can be employed to depict *program transformations*. A program transformation $\mathcal{T}$ can be defined as a triple $\langle \mathcal{S}_{in}, \mathcal{S}_{out}, \mathcal{R} \rangle$, where $\mathcal{S}_{in}$ is an input program schema describing the context where the transformation should be applied, $\mathcal{S}_{out}$ is an output program schema, describing how the input program should be modified by the transformation and $\mathcal{R}$ is a set of constraints for both schemata. The intuitive meaning of a transformation $\mathcal{T}$ is "if program Π matches $\mathcal{S}_{in}$ under semi-unification Θ, subject to $\mathcal{R}$, then obtain a new program Π' applying the same semi-unification Θ to $\mathcal{S}_{out}$, also subject to $\mathcal{R}$". We can formally define a *transform* relation between a program Π, a transformation $\mathcal{T}$ and a transformed program Π':

$$transform(\Pi, \langle \boxed{\begin{array}{c|c} \mathcal{S}_{in} & \mathcal{S}_{out} \\ \hline \multicolumn{2}{c}{\mathcal{R}} \end{array}} \rangle, \Pi') \leftarrow \delta(\mathcal{S}_{in}, \Pi, \Theta) \ \wedge \ \lceil \mathcal{S}_{in}\Theta = \Pi \rceil^{\mathcal{R}} \ \wedge \delta(\mathcal{S}_{out}, \Pi', \Theta)$$

That is, relation *transform* holds between a transformation $\mathcal{T}$ (shown with its components enclosed in boxes, with its schemata side by side and its constraints underneath) and two programs Π and Π', if $\mathcal{S}_{in}$ semi-unifies with Π through Θ subject to $\mathcal{R}$ and $\mathcal{S}_{out}$ semi-unifies with Π' through the same Θ also subject to $\mathcal{R}$. We show in Figure 3 an example of a transformation for logic programs. This transformation formalises an opportunity to improve the program efficiency

$$\mathcal{T}_2 = \langle \boxed{\begin{array}{l|l} \boldsymbol{C}_1 & \boldsymbol{C}_1 \\ S(\boldsymbol{A}_1) \leftarrow \boldsymbol{G}_1, P([x], y, z), \boldsymbol{G}_2 & S(\boldsymbol{A}_1) \leftarrow \boldsymbol{G}_1, z = [x|y], \boldsymbol{G}_2 \\ \boldsymbol{C}_2 & \boldsymbol{C}_2 \\ \hline \multicolumn{2}{l}{P =_{def} \boxed{\begin{array}{l} P([\,], x_1, x_1) \leftarrow \\ P([x_2|x_3], x_4, [x_2|x_5]) \leftarrow P(x_3, x_4, x_5) \end{array}}} \end{array}} \rangle$$

Fig. 3. Sample Transformation for Logic Programs

by elimination of a call to an auxiliary predicate P whose definition is given as a constraint. P corresponds to predicate *append*/3 which appends two lists [2]. When that predicate is called with a singleton list as its first argument, we can safely replace that call by an explicit unification appending the single element of the list. In the transformation above, rather than restricting $\mathcal{S}_{in}$ (left superior box) by specifying the name of P, we confer more generality on the transformation by specifying the definition of P. This allows for variants

of names of predicates to be safely disregarded, as long as its definition is that specified in the constraint.

The formalisation presented here is offered to experienced programmers as a tool to prepare a repertory of transformations like the ones shown above. The transformations should address commonly occurring inefficient constructions and show how they can be changed to improve their computational behaviour. Those desirable properties of the original programs such as termination and declarative meaning should be kept. The person who proposes transformations is responsible for ensuring that these properties are maintained. In [9] we find an investigation on the correctness of transformation schemata.

It should be pointed out that there is a fundamental difference between the characterisation of transformations presented here and those presented in [13]. The transformations shown there address only predicates and not complete programs. As a result, the *context* of the predicates has to be included in the transformation, that is, how they are employed in a program.

5 Applying Transformations

We have in mind a specific setting for applying those transformations written in our schema language with constraints: given a program, an automated system will try to use every possible transformation in it following a pre-defined analysis strategy. The transformations are then said to be *opportunities* to improve programs and our system that tries to use them is hence said to be *opportunistic*.

Our analysis and transformation framework is completely automatic: a given logic program is exhaustively and methodically analysed and all possible transformations are applied to it. The program initially supplied is analysed and, if possible, one transformation is applied to it; the resulting program is fed back into the system and is, on its turn, analysed. This process eventually converges when a *stable* program (optimal with respect to the set of available transformations) is obtained, to which no further transformations are applicable. Our analysis framework can be generically defined as the following recursive relation:

$$\mathit{apply\_transf}(\Pi, \Pi^{Final}) \leftarrow \mathit{transform}(\Pi, \mathcal{T}, \Pi') \ \wedge \ \mathit{apply\_transf}(\Pi', \Pi^{Final})$$
$$\mathit{apply\_transf}(\Pi, \Pi^{Final}) \leftarrow \Pi^{Final} = \Pi$$

That is, relation *apply_transf* between programs Π and Π^{Final} holds if we apply one transformation $\mathcal{T}$ to Π yielding intermediate program Π', and Π' and Π^{Final} are recursively related (first case) via *apply_transf*. If no transformations are applicable to Π (second case), then $\Pi = \Pi^{Final}$. The termination/convergence of *apply_transf* is determined by the repertory of available transformations. If the prescribed alterations of a transformation are undone by another transformation (or a sequence of them), then *apply_transf* may loop forever, with the program being altered and then changed back into its original form. Transformations should, therefore, be designed with its intended use in mind, that is, the *apply_transf* framework. The designer of a transformation ought to take into account that it will be exhaustively matched against a program and, whenever possible, its prescribed modifications will be carried out.

5.1 Choosing a Transformation: Heuristics

It is possible that at one moment there is more than one applicable transformation to a program Π. In order to solve such an impasse, we adopt the same approach of [13] and assign heuristic values to each transformation. Given a set of transformations applicable to a program, we will choose among them that transformation with the best heuristic value. A transformation $\mathcal{T}$ is added with one more component, its heuristic value h, and it will be represented as the 4-uple $\langle h, \mathcal{S}_{in}, \mathcal{S}_{out}, \mathcal{R} \rangle$.

We also adopt the rationale described in [13] to assign heuristic values to transformations: a transformation should be assessed for its potential subsequent use, maximising the number of applicable transformations to a program. The underlying hypothesis of such heuristic values is that the more transformations are applied to a program the better it will become. These heuristic values should allow the application of the largest number of transformations, delaying the use of those that may block or impede the subsequent application of other transformations. Such interferences are due to the extension and type of syntactic changes performed on the program which may prevent the match with other transformations. The heuristic values are assigned to transformations based on their potential for interleaving: the less dramatic the changes performed by a transformation, the better should be its heuristic value because its changes will not impede the application of other transformations.

5.2 Automatic Analysis of Logic Programs

In [13] an inefficient exhaustive form for the analysis of logic programs is proposed and embedded into their prototypical system: a computational overhead is caused by the unnecessary analysis of stable subprograms. In [11] a more efficient way to perform the analysis of programs is proposed in which a program is examined starting from its simpler self-contained subprograms and going through more complex subprograms which make use of the already analysed subprograms. This form of analysis is named *bottom-up* and it was employed in their prototypical system **OpTiSB** (**Op**portunistic **T**ransformat**i**on **S**ystem – **B**ottom-Up). In this work we choose the portions of program for analysis following this bottom-up fashion, it also being incorporated to our prototype, **OpTiSC** (**Op**portunistic **T**ransformat**i**on **S**ystem with **C**onstraints), which employs the schema language presented here.

5.3 Working Example

We show on the left-hand side of Figure 4 a Prolog program comprising three predicates. Our approach for program analysis and transformation initially considers *rev*/2 (it holds if its second argument is a list with its elements in the reverse order of the list in its first argument) and *sum*/2. This is due to our adopted bottom-up policy: the simplest self-contained subprograms are analysed, then those immediately larger subprograms which employ the already analysed and stable subprograms, and so on (we assume, for the sake of this example and

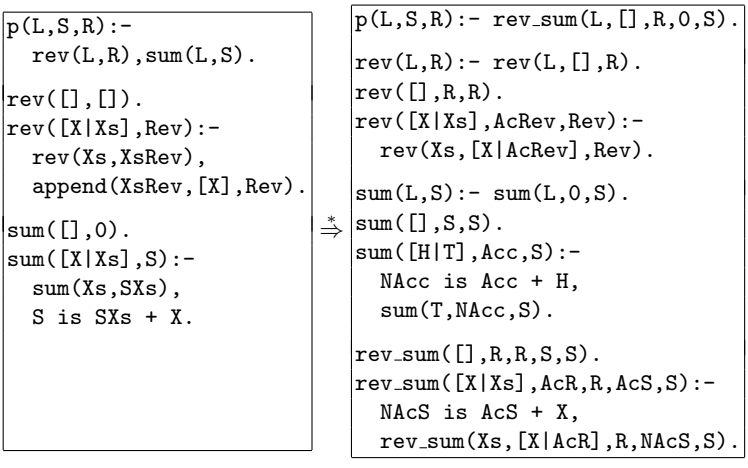


Fig. 4. Initial Program (Left) and its Stable Transformed Version

without loss of generality, that *append*/3 is a system built-in predicate). The original version of $p/3$ (it finds the reverse R of the input list L and the sum S of its elements) processes the input list twice, once to find its reverse and once to sum its elements. A suitable transformation (shown in Figure 5) can merge

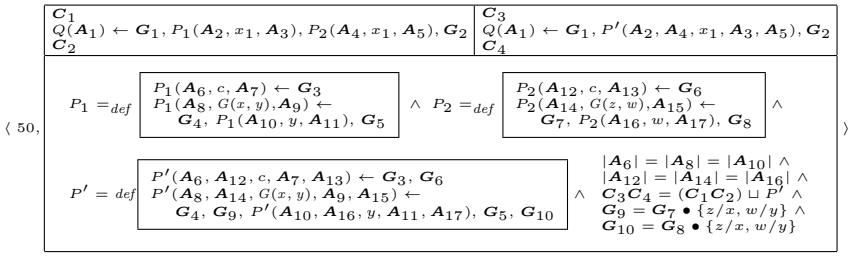


Fig. 5. Transformation for Merging Predicates Processing Same Data Structure

rev/2 and *sum*/2 into a single predicate *rev_sum*/5 which processes the list only once, performing both the reversal and the sum. Therefore the merged predicates are the newly obtained versions with their accumulator pairs. The $\stackrel{*}{\Rightarrow}$ indicates the exhaustive application of *apply_transform* – the final program is *stable* in the sense that it cannot be further altered by any transformation. If the right-hand side program is fed back into our system, it will be output unchanged: the program is *optimal* with respect to the repertoire of transformations.

6 Conclusions, Comments, and Further Work

In this work we have proposed a schematic language to economically represent classes of logic programs, extending the work of [13]. We have added to our schema language the constraint extension inspired in [3], but we have adapted

it so as not to cause exponential costs during a matching. We have defined the syntax and semantics of our language in terms of the actual logic programming constructs they stand for. We have explained the concepts of semi-unification, abstraction and specialisation and how they are related and we have shown, in a declarative fashion, how to obtain semi-unifications between a program and a schema (the definition of δ). We have also shown how our schematic notation can be used to represent useful and practical logic program transformations. This proposal is aimed at an automated context (also formally defined here) for opportunistic program analysis: whenever an opportunity to apply a transformation (depicted in our language) has been detected then the prescribed changes will be carried out. Our schema language with constraints should be used with the context of its application in mind, otherwise the termination of the analysis is jeopardised.

In our language, we make a difference between program variables and terms using appropriate meta-variables x and t to respectively represent them. In [13] and [3], however, a first-order variable of a schema can be associated with any term of a program, that is, variables, constants or any other compound term. Such semantic for the schematic construct is, in our opinion, confused with the role of variables in logic programming and should be avoided. It should also be pointed out that in our schema language the permutation of program constructs cannot be represented, as in the proposal of [3]. This restriction makes the semi-unification process computationally feasible, with an algorithm of polynomial complexity. However, there is no loss in the expressiveness because we can still prepare different program schemata to explicitly describe the possible permutations.

Another important contribution of this work concerns the expressiveness of program transformations represented in our formalism. Previous work [13] only addressed isolated predicates and the application of a transformation was followed by post-processing and updating of the complete program. Our transformations, contrastingly, contemplate complete programs and their application need not resort to any extraneous procedures.

Our approach can be seen as a Term Rewriting System (TRS) [14], in which the input schema is the left side of a rewriting rule and the output schema is the right side of it. Therefore, our system is susceptible to the same problems related with TRS's, such as infinite loops and non-convergences. The proponent of the transformations should make sure that they do not interfere with each other in a negative way, undoing (directly or not) their prescribed alterations. When transformations are proposed, we should pay special attention to the maintenance of the program's declarative meaning (that is, the same and all answers should be computed after the program has been modified). In such circumstances, the context in which the transformation should be applied has to be precisely identified so as to state precisely those cases in which it should be used and to leave out those other cases where it should not. Our repertoire of constraints provide an ideal way to represent the context, with its concise syntax and precise meaning.

The available transformations should keep unaltered the declarative meaning of the programs to which they are applied. Currently all responsibility to obey this restriction rests on the person who designs the transformation. It would be

useful, though, to provide users with a method to define (restricted) classes of transformations which were guaranteed to keep the declarative meaning of programs unchanged along the lines of [9]. Indeed the preparation of transformations is a task much in need for automated support. The preparation of useful transformations (that is, of general applicability and of practical importance) is an activity that involves ingenuity and creativity. Moreover, since the transformations are usually represented in an abstract formalism in which classes of programs are contemplated, their design becomes a difficult and error-prone activity. We suggest in [12] an approach to help designers to propose transformations from concrete examples of programs.

References

1. H. Aït-Kaci. *Warren's Abstract Machine*. MIT Press, USA, 1991.
2. Krzysztof R. Apt. *From Logic Programming to Prolog*. Prentice-Hall, U.K., 1997.
3. E. Chasseur and Y. Deville. Logic Program Schemas, Constraints and Semi-Unification. In *LNCS, Vol. 1463*. Springer, 1998.
4. N. E. Fuchs and M. P. J Fromherz. Schema-Based Transformations of Logic Programs. In *Proc. of LoPSTr'91*. Springer, 1992.
5. G. Huet and B. Lang. Proving and Applying Program Transformations Expressed with Second-Order Patterns. *Acta Informatica*, 11:31–55, 1978.
6. F. Nielson. Perspectives on Program Analysis. *ACM Comp. Surv.*, 28(4es), 1996.
7. R. Paige. Future Directions in Program Transformation. *ACM Comp. Surv.*, 28(4es), 1996.
8. M. Proietti and A. Pettorossi. Transformations of Logic Programs: Foundations and Techniques. *J. Logic Progr.*, 19, 20:261–320, 1994.
9. J. D. C. Richardson and N. E. Fuchs. Development of Correct Transformation Schemata for Prolog Programs. In *LNCS, Vol. 1463*. Springer, 1998.
10. L. Sterling and E. Shapiro. *The Art of Prolog*. MIT Press, 1986.
11. W. W. Vasconcelos, M. A. Aragão, and N. E. Fuchs. Automatic Bottom-Up Analysis and Transformation of Logic Programs. In *LNAI, Vol. 1159*. Springer, 1996.
12. W. W. Vasconcelos and M. A. T. Aragão. Automatic Extraction of Logic Program Transformations from Examples. Tech. Rep., 1999. Available upon request.
13. W. W. Vasconcelos and N. E. Fuchs. An Opportunistic Approach for Logic Program Analysis and Optimisation using Enhanced Schema-Based Transformations. In *LNCS, Vol. 1048*. Springer, 1996.
14. J. Zanutto. Term Rewriting Systems: Theory and an Application. MSc Diss., IME. São Paulo Univ., Brazil, May 1999. In Portuguese.

A Schema Language: Syntax & Semantics

We formally define below our schema language using a context-free grammar. Nonterminal symbols are represented in italics starting with capital letter; a stands for a generic constant symbol, x for a generic variable symbol, p a predicate symbol, f a functor symbol, P a generic predicate variable, F a generic functional variable and $\top$ the empty literal.

$$
\begin{aligned}
\mathit{ProgSch} &\rightarrow \mathit{ClauseSch}\ \mathit{ProgSch} \mid \top \\
\mathit{ClauseSch} &\rightarrow \boldsymbol{C}_i \mid \mathit{HeadSch} \leftarrow \mathit{BodySch} \\
\mathit{HeadSch} &\rightarrow p(\mathit{TrmSchmta}) \mid P(\mathit{TrmSchmta}) \mid p \mid P \\
\mathit{BodySch} &\rightarrow \mathit{LiteralSch}, \mathit{BodySch} \mid \top \\
\mathit{LiteralSch} &\rightarrow \boldsymbol{G}_i \mid \mathit{HeadSch} \\
\mathit{TrmSchmta} &\rightarrow \mathit{TrmSch}, \mathit{TrmSchmta} \mid \mathit{TrmSch} \\
\mathit{TrmSch} &\rightarrow \boldsymbol{A}_i \mid f(\mathit{TrmSchmta}) \mid F(\mathit{TrmSchmta}) \mid a \mid x \mid t
\end{aligned}
$$

We also define here the meaning of our schematic notation by relating them with actual logic programming constructions via the δ operator. However, we need to initially define the class of logic programs we want to address – the following context-free grammar formally characterises it:

$$
\begin{aligned}
\mathit{Prog} &\rightarrow \mathit{Clause}\ \mathit{Prog} \mid \top & \qquad \mathit{Body} &\rightarrow \mathit{Literal}, \mathit{Body} \mid \top \\
\mathit{Clause} &\rightarrow \mathit{Literal}\ \texttt{:-}\ \mathit{Body} & \qquad \mathit{Trms} &\rightarrow \mathit{Trm}, \mathit{Trms} \mid \mathit{Trm} \\
\mathit{Literal} &\rightarrow p(\mathit{Trms}) \mid p & \qquad \mathit{Trm} &\rightarrow f(\mathit{Trms}) \mid a \mid x
\end{aligned}
$$

Given the grammars above we define below $\delta(\mathit{ProgSch}, \mathit{Prog}, \Theta)$ mapping their languages. Initially, we have the base case $\delta(\top, \top, \emptyset)$, that is, an empty program schema and an empty program are related through an empty semi-unification, and $\delta(\boldsymbol{C}, \mathit{Prog}, \{\boldsymbol{C}/\mathit{Prog}\})$, that is, a vector of clauses relates with any logic program through semi-unification $\{\boldsymbol{C}/\mathit{Prog}\}$. We show below the generic case for the relationship between program schemata and programs:

$$
\begin{aligned}
\delta(\mathit{ProgSch}, \mathit{Prog}, \Theta) \leftarrow\ & \mathit{ProgSch} = \mathit{ProgSch}_1 \cdot \mathit{ProgSch}_2 \wedge \mathit{Prog} = \mathit{Prog}_1 \cdot \mathit{Prog}_2 \wedge \\
& \delta(\mathit{ProgSch}_1, \mathit{Prog}_1, \Theta_1) \wedge \delta(\mathit{ProgSch}_2, \mathit{Prog}_2, \Theta_2) \wedge \Theta = \Theta_1 \cup \Theta_2
\end{aligned} \tag{1}
$$

That is, *ProgSch* is related with *Prog* via Θ if we decompose them respectively as $\mathit{ProgSch}_1$ and $\mathit{ProgSch}_2$ and Prog_1 and Prog_2 and these decompositions are, on their turn, correspondingly related. Semi-unification Θ is the union of the semi-unifications between the corresponding subparts of the schema and program. We show below the relationship between schematic and program clauses:

$$
\begin{aligned}
\delta(\mathit{ClauseSch}, \mathit{Clause}, \Theta) \leftarrow\ & \mathit{ClauseSch} = (\mathit{HeadSch} \leftarrow \mathit{BodySch}) \wedge \mathit{Clause} = (\mathit{Literal}\ \texttt{:-}\ \mathit{Body}) \wedge \\
& \delta(\mathit{HeadSch}, \mathit{Literal}, \Theta_1) \wedge \delta(\mathit{BodySch}, \mathit{Body}, \Theta_2) \wedge \Theta = \Theta_1 \cup \Theta_2
\end{aligned} \tag{2}
$$

That is, *ClauseSch* is related with *Clause* through Θ if there is a relationship between the corresponding parts of them – the schematic head *HeadSch* should be related with the actual head *Literal* and the schematic body *BodySch* with the actual body *Body*. Θ is the union of the semi-unifications for these relationships. We describe below the relationship between sequences of literals and their schemata.

$$
\begin{aligned}
\delta(\mathit{BodySch}, \mathit{Body}, \Theta) \leftarrow\ & \mathit{BodySch} = \mathit{BodySch}_1 \cdot \mathit{BodySch}_2 \wedge \mathit{Body} = \mathit{Body}_1 \cdot \mathit{Body}_2 \wedge \\
& \delta(\mathit{BodySch}_1, \mathit{Body}_1, \Theta_1) \wedge \delta(\mathit{BodySch}_2, \mathit{Body}_2, \Theta_2) \wedge \Theta = \Theta_1 \cup \Theta_2
\end{aligned} \tag{3}
$$

This relationship is similar to formula 1, but here it is adapted for sequences of literals and their schemata. The semi-unification between specific literals and their schemata is given by $\delta(\boldsymbol{G}, \mathit{Body}, \{\boldsymbol{G}/\mathit{Body}\})$, that is, a vector of literals $\boldsymbol{G}$ semi-unifies with any sequence of literals *Body* (including, of course, the empty literal $\top$). Other possibilities are:

$$
\delta(P(\mathit{TrmSchmta}), p(\mathit{Trms}), \Theta) \leftarrow \delta(\mathit{TrmSchmta}, \mathit{Trms}, \Theta') \wedge \Theta = \Theta' \cup \{P/p\} \tag{4}
$$

$$
\delta(p(\mathit{TrmSchmta}), p(\mathit{Trms}), \Theta) \leftarrow \delta(\mathit{TrmSchmta}, \mathit{Trms}, \Theta) \tag{5}
$$

Formula 4 describes the semi-unification between $P(\mathit{TrmSchmta})$ and $p(\mathit{Trms})$: it will be the union of the semi-unification between *TrmSchmta* and *Trms* and $\{P/p\}$. When the literal schema does not have a predicate variable then it only semi-unifies with an actual literal if there is a syntactic correspondence between their predicate symbols p – their semi-unification will be that between their schematic terms and actual terms. Further cases are $\delta(P, p, \{P/p\})$ and $\delta(p, p, \emptyset)$. We show below the relationship between sequences of schematic terms and actual terms.

$$\delta(TrmSchmta, Trms, \Theta) \leftarrow TrmSchmta = TrmSchmta_1 \cdot TrmSchmta_2 \wedge Trms = Trms_1 \cdot Trms_2 \wedge \delta(TrmSchmta_1, Trms_1, \Theta_1) \wedge \delta(TrmSchmta_2, Trms_2, \Theta_2) \ \wedge \ \Theta = \Theta_1 \cup \Theta_2 \quad (6)$$

This definition is an adaptation of Formula 1 for sequences of schematic terms and actual terms. We explore below the semi-unification between a schematic term and an actual program term:

$$\delta(F(TrmSchmta), f(Trms), \Theta) \leftarrow \delta(TrmSchmta, Trms, \Theta') \wedge \Theta = \Theta' \cup \{F/f\} \quad (7)$$

$$\delta(f(TrmSchmta), f(Trms), \Theta) \leftarrow \delta(TrmSchmta, Trms, \Theta) \quad (8)$$

Further (base) cases are $\delta(\boldsymbol{A}, Trms, \{\boldsymbol{A}/Trms\})$, $\delta(t, Trm, \{t/Trm\})$, $\delta(a, a, \emptyset)$ and $\delta(x, y, \{x/y\})$.

A.1 Semantics of Constraints

We formally define here the meaning of our constraints. Our first definition below relates schemata, semi-unifications, programs and sets of constraints:

$$\lceil \mathcal{S} \cdot \Theta = \Pi \rceil^{\mathcal{R}} \leftarrow \mathcal{R} = \{R_0 \wedge \cdots \wedge R_n\} \wedge \delta(\mathcal{S}, \Pi, \Theta) \wedge \sigma(R_0, \Theta) \wedge \cdots \wedge \sigma(R_n, \Theta) \quad (9)$$

That is, the relationship holds if each element R_i of the possibly empty set $\mathcal{R}$ of constraints satisfies auxiliary relation σ, defined below. The σ relation checks whether a constraint holds true for a given semi-unification, and its definition formalises the meaning of each constraint. The size constraints hold if the following is true:

$$\sigma(|SchVec| \circ n, \Theta) \leftarrow SchVec/ProgCnstr \in \Theta \ \wedge \ |ProgCnstr| \circ n \quad (10)$$

That is, the constraint holds if *ProgCnstr* associated with *SchVec* in Θ satisfies the $\circ\, n$ size restriction. Content constraints are defined as:

$$\sigma(\langle SchComp_1, \ldots, SchComp_n\rangle \sqsubseteq Vector, \Theta) \leftarrow Vector/ProgCnstr \in \Theta \wedge ProgCnstr = ProgCnstr_1 \cdot ProgCnstr_2 \cdot ProgCnstr_3 \wedge \delta(\langle SchComp_1, \ldots, SchComp_n\rangle, ProgCnstr_2, \Theta) \quad (11)$$

That is, $\langle SchComp_1, \ldots, SchComp_n\rangle$ should be semi-unified with $ProgCnstr_2$ of *ProgCnstr* (associated with *Vector*) in Θ. Difference and union constraints are defined as:

$$\sigma(SchProg_1 = SchProg_2 - \{C_1, \ldots, C_n\}, \Theta) \leftarrow SchProg_1/Prog_1 \in \Theta \wedge SchProg_2/Prog_2 \in \Theta \wedge C_1/AcC_1 \in \Theta \wedge \cdots \wedge C_n/AcC_n \in \Theta \wedge Prog_1 = Prog_2 - \{AcC_1, \ldots, AcC_n\} \quad (12)$$

$$\sigma(SchProg_1 = SchProg_2 \sqcup SchProg_3, \Theta) \leftarrow SchProg_1/Prog_1 \in \Theta \wedge SchProg_2/Prog_2 \in \Theta \wedge SchProg_3/Prog_3 \in \Theta \wedge Prog_1 = Prog_2 \sqcup Prog_3 \quad (13)$$

That is, the difference and union constraints hold if appropriate relationships hold among the actual program constructs with which the program schemata are associated in Θ. Definition constraints have their meaning formalised as:

$$\sigma(P =_{def} SchProg, \Theta) \leftarrow \delta(SchProg, Prog, \Theta) \wedge Prog = \langle P(Terms_0) \text{ :- } Body_0, \ldots, P(Terms_m) \text{ :- } Body_m\rangle \quad (14)$$

That is, P is defined as *SchProg* if the clauses associated with *SchProg* in Θ all have predicate symbol P (same name and arity) as their head goals. Finally, we define the meaning of the replacement constraints as

$$\sigma(SchVec_1 = SchVec_2 \bullet \{x_1/Term_1, \ldots, x_n/Term_n\}, \Theta) \leftarrow SchVec_1/AcC_1 \in \Theta \wedge SchVec_2/AcC_2 \in \Theta \wedge AcC_1 = AcC_2|^{x_1}_{Term_1}|^{x_2}_{Term_2}\cdots|^{x_n}_{Term_n} \quad (15)$$

That is, it holds if AcC_1 (associated with $SchVec_1$) is equal to AcC_2 (associated with $SchVec_2$) if we replace in AcC_2 all occurrences of x_i by $Term_i$.

Experiments in Answer Sets Planning
(Extended Abstract)

M. Balduccini, G. Brignoli, G.A. Lanzarone, F. Magni, and A. Provetti*

Centro di ricerca "Informatica Interattiva"
Università degli Studi dell'Insubria a Varese, Italy.
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano, Italy.

Abstract. The study of formal nonmonotonic reasoning has been motivated to a large degree by the need to solve the frame problem and other problems related to representing actions. New efficient implementations of nonmonotonic reasoning, such as SMODELS and DLV, can be used to solve many computational problems that involve actions, including plan generation. SMODELS and its competitors are essential to implement a new approach to knowledge representation and reasoning: to compute solutions to a problem by computing the stable models (answer sets) of the theory that represents it. Marek and Truszczyński call this paradigm Stable model programming. We are trying to assess the viability of stable logic programming for agent specification and planning in realistic scenarios. To do so, we present an encoding of plan generation within the lines of Lifschitz's Answer set planning and evaluate its performance in the simple scenario of Blocks world. Several optimization techniques stemming from mainstream as well as satisfiability planning are added to our planner, and their impact is discussed.

1 Introduction

Stable Logic Programming (SLP) is an emergent, alternative style of logic programming: each solution to a problem is represented by an answer set of a function–free logic program[1] encoding the problem itself. Several implementations now exist for stable logic programming, and their performance is rapidly improving; among them are SMODELS [Sim97,NieSim98], DERES [ChoMarTru96], SLG [CheWar96], DLV [ELMPS97], and CCALC [McCTur97].

Recently, Lifschitz has introduced [Lif99] the term Answer set planning to describe which axioms are needed to characterize correct plans and to discuss the applicability of stable model (answer set) computation interpreters to plan generation. Two features of Answer set planning are particularly attractive.

First, the language it uses is very apt to capture planning instances where conventional STRIPS fall short. In fact, Answer set planning allow the user to specify incomplete knowledge about the initial situation, actions with conditional and indirect effects, nondeterministic actions, and interaction between concurrently executed actions.

* Corresponding author, ph: +39-02-55006.290, e-mail: provetti@dsi.unimi.it.

[1] Or, via a syntactic transformation, a restricted default theory or even a $DATALOG^{\neg}$ program.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 95–107, 2000.

Second, even tough Answer set planning specifications, like those in this paper, are completely declarative in nature, SMODELS is by now efficient enough to interpret them, i.e., generate valid plans, with times at least comparable to those of on-purpose planners, on which research and optimization has been active for years. For instance, Dimopoulos et al. [DNK97] report that on the *logistics* domains from Kautz and Selman [KauSel96] (which is about package delivery by using trucks and planes) their SMODELS solution took 18 seconds vis-a-vis with more than 6 hours for GRAPHPLAN.

We would like to develop on Lifschitz's proposal in several directions which are related to, and bring together, our current research in nonmonotonic reasoning and autonomous agents.

Research in Milan [Cos95,BCP99,CosPro99], has so far contributed to stable logic programming by analyzing, in particular, the dependence of stable models existence on the syntax of the program. This brought out [Cos95] several basic results that extend the usual sufficent conditions of [Dun92,Fag94], none of which indeed applies to Answer set planning, as well as suggesting a new stable model computation algorithm, proposed in [BCP99].

Meanwhile, research in Varese has discussed the prospect of applying logic programming to the complex tasks of designing and *animating* an autonomous agent capable of reaction, planning and above all learning an unknown outside environment. A PROLOG program simulating a learning agent and its environment was proposed in [BalLan97]. Now, a physical scenario is being considered, with the goal of implementing a controller for a small mobile robot equipped with sensors and an arm. For a *principled* knowledge representation approach to succeed there, we need a viable computational mechanism, and this paper is the first of a series of experiments towards assessing stable models computation as a valid choice.

This article discusses a set of experiments on the problem of plan generation in a blocks world domain. This is a simple, familiar problem from the literature, which is often taken as a benchmark for comparing planners. Planning in domains amenable of, basically, a propositional fluents encoding has a natural representation in terms of stable models computations. Some of the reasons are:

- encodings are rather concise and easy to understand;
- the cost of generating a plan grows rapidly with the number of blocks considered and
- the type of planning needed for our autonomous robot is not too distant from that considered here.

Our experiments have been carried out using SMODELS , one successful implementation of stable models programming. For our project, the appealing feature of SMODELS over its competitors is the companion grounding program LPARSE, which accepts as input programs with variables, [some types of]functions symbols and constraints, intended as notation shorthands[2].

[2] Technically, this means that the Herbrand universe of the programs remains finite; practically, it is described as a preprocessing phase, carried out by LPARSE that removes functional terms in favor of internally-constructed constants.

2 Background Definitions

The answer sets semantics [GelLif88,GelLif91] is a view of logic programs as sets of inference rules (more precisely, default inference rules), where a stable model is a set of literals closed under the program itself. Alternatively, one can see a program as a set of constraints on the solution of a problem, where each answer set represents a solution compatible with the constraints expressed by the program. Consider the simple program $\{q \leftarrow not\ p, not\ c.\ \ p \leftarrow not\ q.\ p \leftarrow c.\}$. For instance, the first *rule* is read as "assuming that both p and c are false, we can *conclude* that q is true." This program has two answer sets. In the first, q is true while p and c are false; in the second, p is true while q and c are false. When all literals are positive, we speak in terms of *stable models.* In this paper we consider, essentially, the language $DATALOG^{\neg}$ for deductive databases, which is more restricted than traditional logic programming. As discussed in [MarTru99], this restriction is not a limitation at this stage.

A rule ρ is defined as usual, and can be seen as composed of a conclusion $head(\rho)$, and a set of conditions $body(\rho)$, the latter divided into positive conditions $pos(\rho)$ and negative conditions $neg(\rho)$. Please refer to [AptBol94] for a thorough presentation of the syntax and semantics of logic programs. For the sake of clarity however, let us report the definition of stable models. We start from the subclass of positive programs, i.e. those where, for every rule ρ, $neg(\rho) = \emptyset$.

Definition 1. *(Stable model of positive programs)*
The stable model $a(\Pi)$ *of a positive program* Π *is the smallest subset of* $\mathbb{B}_\Pi$ *such that for any rule* $a \leftarrow a_1, \ldots a_m$ *in* Π*:* $a_1, \ldots, a_m \in a(\Pi) \Rightarrow a \in a(\Pi)$.

Positive programs are unambiguous, in that they have a unique stable model, which coincides with that obtained applying other semantics.

Definition 2. *(Stable models of programs)*
Let Π *be a logic program. For any set* S *of atoms, let* Π^S *be a program obtained from* Π *by deleting (i) each rule that has a formula 'not* A*' in its body with* $A \in S$*, and (ii) all formulae of the form 'not* A*' in the bodies of the remaining rules.*

Π^S *does not contain "not," so that its stable model is already defined. If this stable model coincides with* S*, then we say that* S *is a* stable model *of* Π*. In other words, the stable models of* Π *are characterized by the equation:* $S = a(\Pi^S)$.

The answer set semantics is defined similarly by allowing the unary operator $\neg$, called explicit negation, to distinguish it from the classical-logic connective. What changes is that we do not allow any two contrary literals a, $\neg a$ to appear in an answer set.

Gelfond and Lifschitz [GelLif91] show how to *compile away* explicit negations by i) introducing extra atoms $a', b' \ldots$ to denote $\neg a, \neg b, \ldots$ and ii) considering only stable models of the resulting program that contain no contrary pair a, a'. This requirement is captured by adding, for each new atom a', the constraint $\leftarrow a, a'$ to the program. In any case, two-valued interpretations can be forced by adding rules $a \leftarrow not\ a'$ and $a' \leftarrow not\ a.$ for each contrary pair of atoms (resp. literals).

2.1 Consistency Conditions

Unlike with other semantics, a program may have no stable model (answer set), i.e., be contradictory, like the following: $\{a \leftarrow not\, b.\ b \leftarrow not\, c.\ c \leftarrow not\, a.\}$, where no set of literals is closed under the rules. Inconsistency may arise, realistically, when programs are combined: if they share atoms, a subprogram like that above may *surface* in the resulting program.

In the literature, the main (sufficient) condition to ensure the existence of stable models is call–consistency [Dun92], which is summarized as follows: no atom *depends* on itself via an odd number of negative conditions. This condition is quite restrictive, e.g., it applies to almost no program for reasoning about actions and planning seen in the literature (see the examples in [Lif99]). Indeed, note how in the translation above the definition of a/a' is not stratified and that the consistency constraint is mapped into rule $false \leftarrow a, a', not\ false$, which is not call-consistent either.

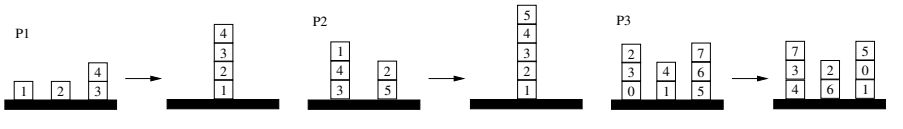
However, some of these authors have shown that this feature does not compromise program's consistency for a large class of cases. The *safe cycle* condition of [CosPro99] applies to all programs considered here.

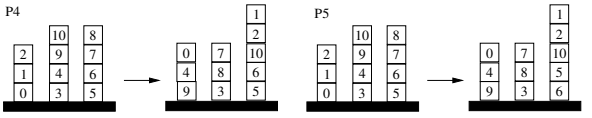
3 Plan Specification

The formalization of the Blocks world as a logic program is the main example used by Lifschitz [Lif99] for introducing answer set planning. Two implementations have stem from Lifschitz's definition, Erdem's [Erd99] and the one introduced hereby.

Erdem's solution, which is the closest to Lifschitz's axioms, uses *action and fluent atoms* indexed by time, i.e., *on(B,B1,T)* is a fluent atom, read as "block B is on block B1 at time T" while *move(B,L,T)* is an action, read as "block B is moved on location (meaning another block or the table surface) L at time T."

Our solution is closer to standard situation calculus, since it employs fluent and action terms. In fact, our version of the two atoms presented above is $holds(on(B, B1, T)$ and $occurs(move(B, L), T)$, respectively. Unlike in standard Situation Calculus, we do not have function symbols to denote 'next situations'. This is due in part to earlier limitations of LPARSE, which until recently allowed only functions on integers. However, by having fluents represented as terms, we need only two inertia axioms (see listing below); hence, our solution is more apt than Erdem's for treating complex planning domains, involving hundreds of fluents. The illustrations below are courtesy of W. Faber.





Problem	blocks	steps
P1	4	4
P2	5	6
P3	8	8
P4	11	9
P5	11	11

The price we pay for the convenience of using fluent and action terms is that whereas Erdem's rules can easily be transformed to make them suitable for DLV or CCALC computation, our programs are not easily rephrased for interpreters other than SMODELS . On the other hand, [FabLeoPfe99] have used planning in the Blocks world to experiment with their **dlv** system. However, we felt that this disadvantage was only transient, as DLV and CCALC are actively pursuing the development of their system.

To describe our planner, let us start from the specification of the instance. Basically, we specify the initial and the goal situation[3]. The initial situation here is completely described but it is easy to allow for incomplete knowledge and –if needed– add default assumptions.

```
%%%%%%%%%%%%%%%%% Domain description: P3
block(b0).
block(b1).
[...]
block(b7).

%%%%%%%% initial situation
holds(on(b2,b3),0).
holds(on(b3,b0),0).
holds(on(b0,table),0).
holds(on(b4,b1),0).
holds(on(b1,table),0).
holds(on(b7,b6),0).
holds(on(b6,b5),0).
holds(on(b5,table),0).

holds(top(b2),0).
holds(top(b4),0).
holds(top(b7),0).
holds(neg(top(b3)),0).
holds(neg(top(b0)),0).
holds(neg(top(b1)),0).
holds(neg(top(b6)),0).
holds(neg(top(b5)),0).
```

The goal is given in terms of a constraint to be satisfied *at the latest* at time *t=depth,* where *depth* is either passed with the SMODELS call or assigned within the input instance.

[3] The instance below is from [Erd99].

```
%%%%%%% goal state
:-not goal(depth).

goal(T) :- time(T), holds(on(b7,b3),T),
           holds(on(b3,b4),T),
           holds(on(b4,table),T),
           holds(on(b2,b6),T),
           holds(on(b6,table),T),
           holds(on(b5,b0),T),
           holds(on(b0,b1),T),
           holds(on(b1,table),T).
```

Let us now see the general axioms for the Blocks world domain (some base predicate definitions are omitted but easy to understand by their use).
The next set of rules describes the [direct] effect of actions.

```
holds(on(B,L),T1) :- next(T,T1),
                     block(B),
                     location(L),
                     occurs(move(B,L),T).

holds(top(B),T1) :- next(T,T1),
                    block(B),
                    location(L),
                    occurs(move(B,L),T).

holds(top(B),T1) :- next(T,T1),
                    block(B), block(B1),
                    holds(on(B1,B),T),
                    location(L),
                    occurs(move(B1,L),T).

holds(neg(top(B)),T1) :- next(T,T1),
                         block(B), block(B1),
                         occurs(move(B1,B),T).
```

The next set of actions provide static constraints, i.e., make no reference to next/past state in order to derive the value of fluents at T (`neq` is the built-in inequality test).

```
holds(neg(on(B,L)),T) :- time(T),
                         block(B),
                         location(L),
                         location(L1),
                         neq(L,L1),
                         holds(on(B,L1),T).

holds(neg(on(table,L)),T) :- time(T),
                             location(L).

holds(top(table),T) :- time(T).
```

The action and fluent description part is ended by the inertia axioms. Note the slight simplification of using semi-normal defaults *in lieu* of abnormalities.

```
holds(F,T1) :- fluent(F), next(T,T1), holds(F,T), not holds(neg(F),T1).

holds(neg(F),T1) :- fluent(F), next(T,T1), holds(neg(F),T), not holds(F,T1).
```

The following set of rules, called the control module, is crucial for the performance of the planner. It establishes the fact that in each answer set exactly one action is performed at each time $0 \leq T \leq depth - 1$ (no action is performed at the last time). As a consequence, there are in principle $|\mathcal{A}|^{depth}$ stable models of this set of rules (where $|\mathcal{A}|$ denotes the number of possible actions). This is not the case in practice since we have inserted several extra conditions that avoid generating *hopeless* candidate actions.

```
%%%%%%% Control
occurs(move(B,L),T) :- next(T,T1),
                       block(B),
                       location(L),
                       neq(B,L),        %%% prevents 'moving onto itself'
                       holds(top(B),T), %%% prevents moving a covered block.
                       holds(top(L),T), %%% prevents moving onto an
already-occupied bl.
                       not diff_occurs_than(move(B,L),T).

diff_occurs_than(move(B,L),T) :- next(T,T1),
                                 block(B),
                                 location(L),
                                 block(B1),
                                 location(L1),
                                 occurs(move(B1,L1),T),
                                 neq(B,B1).

diff_occurs_than(move(B,L),T) :- next(T,T1),
                                 block(B),
                                 block(B1),
                                 location(L),
                                 location(L1),
                                 occurs(move(B1,L1),T),
                                 neq(L,L1).
```

The rules above are an application of the nondeterministic choice operator by [SacZan97]. Finally, we have some constraints which further guarantee properties of the plan.

```
%%%%%%%%% Consistency
:- fluent(F),time(T), holds(F,T),holds(neg(F),T).
%%%%%%% Impossibility laws
:- time(T),block(B),location(L),occurs(move(B,L),T),holds(neg(top(B)),T).
:- time(T),block(B),block(B1),occurs(move(B,B1),T),holds(neg(top(B1)),T).
%%%%%%% can't move on the same block where it's already on
:- time(T),block(B),location(L),occurs(move(B,L),T),holds(on(B,L),T).
```

4 Computational Results

As it employs function symbols, the ground version of our planner result much larger than the comparable Erdem's version. As a result, computation time is also longer. The user may or may not want to trade performance for the convenience of using function symbols. The table below reports the timing of finding the minimal plan for each of the Blocks world problems of [Erd99].

Table 1. Times with DEPTH=length of the minimal solution on a Pentium III 500MHz with SunOS 5.7.

Prob.	DEPTH	ERDEM	SITCALC-STYLE
P1	4	0.011	0.100
P2	6	1.480	1.900
P3	8	42.950	1071.300
P4	9	137.560	—
P5	11	—	—

4.1 Linearization

Linearization has been proposed in [KauSel96] to improve performance of SAT-based planning by reducing the number of atoms of a given instance, i.e., essentially, its search space. Even tough the search space of SMODELS is defined differently than that of SATPLAN, viz. it corresponds to literal appearing under negation as failure *only* [Sim97], we have proved that it is very effective also in answer set planning.

The idea is to break up the three-parameters predicate $occurs(move(A, B), T$ into two predicates: $move\_obj(B, T)$ and $move\_dest(L, T)$. Let $|B|$ be the number of blocks and $|\mathcal{A}|$ the number of actions. While *occurs* has $|B|^2 \cdot |T| + |B| \cdot |T|$ instances in the normal case, with linearization we consider only $2|B| \cdot |T| + |T|$ atoms overall.

The changes to be made to our planner are concentrated in the control module, listed below. the renaming changes consist in substituting each occurrence of *occurs* (no pun intended) with either *move_obj* or *move_dest* or both.

```
%%%%%%%%%%%% Linearized control module
move_obj(B,T) :- time(T),
        block(B),
        holds(top(B),T),
        not diff_obj(B,T).

diff_obj(B,T) :- time(T),
        block(B),
        block(B1),
        neq(B,B1),
        move_obj(B1,T).
```

```
diff_obj(B,T) :- time(T),
        block(B),
        not move_obj(B,T).

move_dest(L,T) :- time(T),
        location(L),
        holds(top(L),T),
        block(B),      %
        move_obj(B,T), % cascading choice
        neq(B,L),      %
        not diff_dest(L,T).

diff_dest(L,T) :- time(T),
        location(L),
        location(L1),
        neq(L,L1),
        move_dest(L1,T).
```

The linearized planner has much appealing performance[4]:

Table 2. Times with linearized module and *depth* = length of the minimal solution.

Prob.	DEPTH	LINEAR
P1	4	0.05
P2	6	0.40
P3	8	23.15
P4	9	60.15
P5	11	189.70

What is the trade off here? Linearization makes specifying parallel execution of actions at least awkward [DNK97].

4.2 Overconstraining

Overcostraining is an optimization technique (w.r.t. time) consisting of adding constraint that are logical consequence of the program rules in order to make the search backtrack at earlier points. It has been discussed in [KauSel96] and in the context of SMODELS interpretation by [DNK97]. It is also present in our first planner: it remains easy to check that, apart from consistency, the constraints are subsumed by the extra conditions in the body of *occurs*.

Even tough overconstraining works for optimization of SMODELS interpretation, there are still some doubts about whether it can be applied successfully in all cases. In fact, during the experiments with our planner, we noticed that, by adding certain constraints, we would obtain a dramatic performance improvement (about 50%) without altering the semantics of the program. Of course,

[4] From now on results represent the average over 2 runs on a Pentium II 400MHz with 300MB RAM running NetBSD 1.4.1, SMODELS 2.24 and LPARSE 0.99.20.

overconstraining could explain it, except that *the constraints are now satisfied regardless.* As an example, let us consider the first additional constraint:

```
:- time(T),block(B),location(L),occurs(move(B,L),T),holds(neg(top(B)),T).
```

Since the predicate occurs/2 is not defined in the (linearized) program anymore, the conjunction is false, independently from the other predicates. This means that the constraint can never be applied. Intuitively, this fact should cause, in the best case, a slight increase in the computation time of the program, due to the larger size of the ground instance. On the contrary, we experienced the dramatic performance improvement shown in Table 3. The constraints to which the results refer to are list below:

```
:- time(T),block(B),location(L),occurs(move(B,L),T),holds(neg(top(B)),T).
% Constr. 1

:- time(T),block(B),block(B1),occurs(move(B,B1),T),holds(neg(top(B1)),T).
% Constr. 2

:- time(T),block(B),location(L),occurs(move(B,L),T),holds(on(B,L),T).
% Constr. 3
```

It is evident from the time results that one constraint is enough to produce the performance improvement. The times of the versions using more than one additional constraint seem to be slightly greater than the one-constraint version, but this issue should be further investigated, since the differences are within experimental error.
We have no convincing explanation for the phenomenon described in this section yet.

Table 3. Experimental results on P3 with/without additional constraints.

test type	file name	LPARSE time	SMODELS time	total time
no constraints	no-occurs-noc.p3	0.70s	36.15s	36.85s
constraint 1	no-occurs-noc2.p3	0.70s	22.50s	23.20s
constraint 2	no-occurs-noc4.p3	0.70s	22.20s	22.90s
constraint 3	no-occurs-noc5.p3	0.70s	22.50s	23.20s
constraints 1 and 2	no-occurs-noc3.p3	0.70s	22.10s	22.80s
all constraints	no-occurs.p3	0.70s	22.45s	23.15s

4.3 Improving Performance Further

Experimental results (and common intuition) show that the performance of SMODELS is, roughly, inversely proportional to the size of the ground instance passed to the interpreter. So, it would be good practice to reduce as much as possible the size of the input program by removing any unnecessary rule/atom. On the other hand, adding constraints may effectively speed up the computation by forcing SMODELS to backtrack at an earlier stage.

We found a good solution to this trade-off which applies to our planner and produces about 10% gain on the computational time of SMODELS . The two constraints achieving this improvement are shown below. Their purpose is two prevent the planner from trying to perform any move after the goal is reached.

```
:- time(T),block(B),goal(T),move_obj(B,T).

:- time(T),location(L),goal(T),move_dest(LT).
```

Suppose that, at time $t_0 < depth$, the planner achieves its goal. Since the definitions of the *move_obj* and *move_dest* predicates do not take into consideration the truth of *goal(T)*, a sequence of useless actions would be generated covering times $t \geq t_0$. This approach has several drawbacks. First of all, performing the additional action choices requires computation time, which is, after all, wasted. Second, since the goal was already reached at time t_0, any later action sequence achieves the goal; this means that a large number of models are generated which differ only for the actions performed after reaching the goal.

The set of constraints that we propose simply prevents any action from being performed after the goal has been achieved. The experimental results of the planner with and without the constraints are shown below.

Table 4. Running times for P3 with/without constraints on post-goal actions.

type	file name	LPARSE time	SMODELS time	total time
w/o constraints	no-occurs.p3	0.70s	22.45s	23.15s
with constraints	no-occurs-e.p3	0.70s	20.77s	21.47s

However, this solution does not mean that we are able to capture minimal plan generation within stable logic programming. Deciding whether a program has stable models is an NP-complete problem [MarTru99], while generating a minimal plan is in Δ_2^p [Lib99]. All we can hope to achieve, for minimal planning, it to optimize an algorithm that calls SMODELS as an NP-oracle *at most* a logarithmic number of times. See Liberatore's work ([Lib99] and references therein) for a discussion on these crucial aspects.

Acknowledgments. S. Costantini has greatly motivated us to pursue this work. E. Erdem's work set the framework of our experiments and her lively comments have prompted and motivated us. I. Niemelä and P. Simons have graciously helped us on-line several times with their SMODELS implementation. This research was partially supported by *Progetto cofinanziato MURST "Agenti Intelligenti: interazione ed acquisizione di conoscenza."*

References

AptBol94. Apt, K. R. and Bol, R., 1994. *Logic programming and negation: a survey,* J. of Logic Programming, 19/20.

BalLan97. M. Balduccini and G. A. Lanzarone, 1997. *Autonomous semi–reactive agent design based on incremental inductive learning in logic programming.* Proc. of the ESSLI'97 Symp. on Logical Approaches to Agent Modeling and Design, pages 1–12. Utrecht University.

BarGel94. Baral, C. and Gelfond. M., 1994. *Logic programming and knowledge representation,* J. of Logic Programming, 19/20.

BCP99. Brignoli G., Costantini S. and Provetti A., 1999. *A Graph Coloring algorithm for stable models generation.* Univ. of Milan Technical Report, submitted for publication.

CosPro99. Costantini S. and Provetti A., 1999. *A new method, and new results, for detecting consistency of knowledge bases under answer sets semantics.* Univ. of Milan Technical Report, submitted for publication.

Cos95. Costantini S., 1995. *Contributions to the stable model semantics of logic programs with negation,* Theoretical Computer Science, 149.

CheWar96. Chen W., and Warren D.S., 1996. *Computation of stable models and its integration with logical query processing,* IEEE Trans. on Data and Knowledge Engineering, 8(5):742–747.

ChoMarTru96. Cholewiński P., Marek W. and Truszczyński M., 1996. *Default reasoning system DeReS.* Proc. of KR96, Morgan-Kauffman, pp. 518-528.

DNK97. Dimopoulos Y., Nebel B. and Koehler J., 1997. *Encoding Planning Problems in Nonmonotonic Logic Programs,* Proc. of European Conference on Planning, pp. 169–181.

Dun92. Dung P.M., 1992. *On the Relation between Stable and Well–Founded Semantics of Logic Programs,* Theoretical Computer Science, 105.

ELMPS97. Eiter, T., Leone, N., Mateis, C., Pfeifer, G., and Scarcello, F., 1997. *A deductive system for non-monotonic reasoning.* Proc. Of the 4th LPNMR Conference, Springer Verlag, LNCS 1265, pp. 363–374.

Erd99. Erdem E., 1999. *Application of Logic Programming to Planning: Computational Experiments.* Proc. of LPNMR'99 Conference, LNAI.

FabLeoPfe99. Faber W., Leone N. and Pfeifer G., 1999. *Pushing Goal Derivation in DLP Computations.* Proc. of LPNMR'99 Conference, LNAI.

Fag94. Fages F., 1994. *Consistency of Clark's completion and existence of stable models.* Proc. of 5th ILPS conference.

GelLif88. Gelfond, M. and Lifschitz, V., 1988. *The stable model semantics for logic programming,* Proc. of 5th ILPS conference, pp. 1070–1080.

GelLif91. M. Gelfond and V. Lifschitz., 1991. *Classical negation in logic programs and disjunctive databases.* New Generation Computing, pp. 365–387.

KauSel96. Kautz H. and Selman B., 1996. *Pushing the envelope: Planning, Propositional Logic and Stochastic Search* Proc. of AAAI'96.

Lib99. Liberatore P., 1999. *Algorithms and Experiments on Finding Minimal Models.* Technical Report of University of Rome "La Sapienza."

Lif99. Lifschitz V., 1999. *Answer Set Planning.* Proc. of LPNMR'99 Conference.

MarTru99. Marek, W., and Truszczyński M., 1999. *Stable models and an alternative logic programming paradigm.* The Journal of Logic Programming.

NieSim98. Niemelä I. and Simons P., 1998. *Logic programs with stable model semantics as a constraint programming paradigm.* Proc. of NM'98 workshop. Extended version submitted for publication.

McCTur97. McCain N. and Turner H., 1997. *Causal theories of actions and change.* Proc. of AAAI'97 Conference, pp. 460–465.

SacZan97. Saccà D. and Zaniolo C., 1997. *Deterministic and Non-Deterministic Stable Models.* J. of Logic and Computation.

Sim97. Simons P., 1997. *Towards Constraint Satisfaction through Logic Programs and the Stable Models Semantics,* Helsinki Univ. of Technology R.R. A:47.

SubNauVag95. Subrahmanian, V.S., Nau D., and Vago C., 1995. *WFS + branch and bound = stable models,* IEEE Trans. on Knowledge and Data Engineering, 7(3):362–377.

Competitive Learning Methods for Efficient Vector Quantizations in a Speech Recognition Environment

F. Curatelli and O. Mayora-Ibarra

DIBE – University of Genova
Via Opera Pia 11A, 16145, Genova, Italy
Tel: +39 010 3532287, Fax: +39 010 3532777
franco@dibe.unige.it, oscar@dibe.unige.it.

Abstract. This paper presents a comparison of three competitive learning methods for vector quantizations of speech data in an efficient way. The analyzed algorithms were two batch methods (the Lloyd LBG algorithm and the Neural Gas method) and one on-line technique (K-means algorithm). These methods obtain reduced subsets of codewords for representing bigger data sets. The experiments were designed for speaker dependent and independent tests and consisted in evaluating the reduced training files for speech recognition purposes. In all the studied cases, the results shown a reduction of learning patterns of near 2 orders of magnitude respect to the original training sets without heavily affecting the speech recognition accuracy. The savings in time after using these quantization techniques, made us to consider this reduction results as excellent since they help to approximate the speech matching responses to almost real time. The main contribution of this work refers to an original use of competitive learning techniques for efficient vector quantization of speech data and so, for reducing the memory size and computational costs of a speech recognizer.

1 Introduction

Competitive learning methods have been widely used for solving optimization problems. In this area, a rather large number of models have been proposed with similar goals but with strong differences in the way they work. In general, competitive learning algorithms are used to distribute in an ordered way, a certain number of vectors in high-dimensional spaces. A useful practical application of this spatial distribution of vectors, is related to the selection of relevant patterns for data training. The selection of this significant patterns is done by eliminating useless and redundant information contained in large data sources. These kind of optimizations are frequently reflected in the storing and classification stages with the consequent savings in time and computational resources.

A possible application of this kind of optimizations can be done in the field of Automatic Speech Recognition (ASR). In particular, ASR requires solving many space and time problems that can have a critical effect on the overall

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 108–114, 2000.

system performance [1]. One of these problems, is the selection of relevant data for time and memory optimizations. Usually, speech recognizers have to deal with large data sources containing lots of features and patterns. However, in most of the cases, not all the available information is always relevant for recognition purposes. This carries out the problem of managing unnecessary data that only slows down the learning and response times of the systems without giving any additional information. For this reason, the use of efficient vector quantization systems is of great importance in the data selection stage [2], [3] [4].

This paper presents a comparison among 3 different hard competitive learning methods for data clustering and quantization in a speech recognition application. The comparison is done by using the same speech data for testing the generalized Lloyd algorithm [5], the Neural Gas method [6] and the k-means technique [7]. Each one of these methods are able to obtain different subsets of speech patterns that efficiently represent a bigger data set. The main goal of this work is to identify the similarities and discrepancies of these methods when reducing speech recognition data sets.

2 LBG, Neural Gas, and K-Means Methods

LBG, Neural Gas and K-means, are well known hard competitive learning methods in which input signals determine the adaptation of only single units. These single units constitute the winner at each competitive iteration. In this way, input signals can be presented as finite pattern sets (batch update) or as continuous inputs (on-line adaptations) for obtaining vector clustering and reductions of bigger data sets as shown in figure 1. The LBG or generalized Lloyd algorithm, works as a batch method by adapting in a repeatedly way some random reference vectors to the arithmetic mean of their Voronoi sets [8]. This kind of adaptations are guaranteed to decrease or maintain the distortion error at every iteration. The Neural Gas method's main characteristic, is the cyclic adaptation of some random reference vectors to a specific input according to their distance. It can be demonstrated that this method reaches always convergence when tested with a fixed learning set. In other hand, the K-means method (named because

Fig. 1. Typical 2D space reduction of competitive learning methods.

each reference vector is always the exact arithmetic mean of the input signals), doesn't has a strict theoretical convergence but in practical applications, converges asymptotically to the expected value of its Voronoi region. A detailed explanation of these well know methods can be found in [9] [10].

3 Pattern Construction and Experiments Design

Speech signals are conveniently represented in segmented patterns. For this work, every data pattern was constructed by hamming windowing a speech signal at regular intervals of 10 ms and the resulting segments were processed with a feature extraction tool to obtain 6 perceptual linear prediction (PLP) coefficients [11] for every frame. These 6 dimensional patterns are associated to their correspondent speech unit, contained in a specific phone dictionary. For this application, the dictionary consisted in the 19 possible phones contained in the ten digits when pronounced in Italian language. For example, for the Italian word *uno* (one), the system decomposes it in several segments associating a class number for every frame. The first and last utterances of the word, will be the background noise *bg* (class 1) that exist in the unvoiced fragments before and after the speech start. The second phone will be the *u* sound (class 2), the third will be the *n* (class 3), and so on for all the remaining speech units of the other 9 digits. This classification of phones will lead to a total of 19 different labels for all the possible speech units as shown in table 1.

Table 1. Phone labeling for the ten Italian digits.

Class	Phone	Class	Phone	Class	Phone
1	silence				
2	dz	8	d	14	tt
3	E	9	e	15	ts
4	r	10	t	16	i
5	o	11	k	17	s
6	u	12	w	18	O
7	n	13	a	19	v

The original training sets were constructed by processing several speech files taken from 2 speakers (*anco* and *cabo*). The registrations were made in *Shorten-wav* format [12] taken from the SPK-IRST digit database [13]. Four learning sets (*ancoall, ancoS01, caboall, caboS01*) were constructed using a complete speaker set and a respective subset of it. The *ancoall* and *caboall* learning files were created in 5 sessions, each one with 4 repetitions of the ten digits. This lead us to a total of 200 repetitions for each training set. For the *ancoS01* and *caboS01* files, the sets were constructed considering only one training session. The resumed characteristics of the learning files are listed in table 2.

Table 2. Training sets before pattern reductions.

Training File	Digit repet.	Speech Frames
ancoall	200	14711
ancoS01	40	2849
caboall	200	15691
caboS01	40	3268

In all the cases, the selected features consisted in the 6-dimensional PLP patterns (each one for every 10 ms. frame), as data input for the vector reduction algorithms.

The experiments were done using the mentioned files as input to the vector reduction methods. The clustering process was carried out in all cases considering different fractions of the initial learning files. The stop criteria for the competitive algorithms was set with threshold levels of 1/10 and 1/100 of the original learning sets' frames. This means that in the first case, the reduced pattern set will be only of the 10% of the original data while in the second case the reduction will be done to a 1% level. Such reductions are very useful for a speech recognition application in the sense that lower dimension sets reflect in time and memory savings (essential for real time response achievement and lower confusion in the data).

4 Results

The phone recognition stage is only the first step to further word recognition. Typical values of state of the art phone estimations show values around 60% of accuracy [14] that are enough for obtaining good word recognition (about 98 % of word identification). The performance of the reduced learning files was evaluated with their speech recognition rates when used as training information for the SPEAR (Speech Processing and Recognition) system [15]. The results are presented either for speaker dependent and independent cases and are separated by phone and word recognition. Additional results are also presented when testing the recognizer with already trained patterns.

4.1 Phone Estimation Results

The phone sequence produced by the SPEAR tool was evaluated using an Euclidean distance criteria within a parallel associative memory phone estimation approach [16]. In this kind of method, every test phone is compared with those of the parallel associative memory for selecting the best attractor and so, the correspondent association. The detailed results for the pattern reduction methods are shown in tables 3 and 4. The information illustrates the performance of all the methods under speaker dependent and independent tests, for different data reduction rates.

Table 3. Phone recognition for $\frac{1}{10}$ reduction of learning set.

	OriginalSet	Lloyd	Neural Gas	k-means
Trained Patterns	100%	72.1%	72.1%	72.2%
Speaker Dependen.	62.7%	61.2%	60.8%	58.8%
Speaker Independ.	39.6%	38, 6%	43.6%	49.6%
Mean Results	67.4%	57.27%	58.8%	60.2%

Table 4. Phone recognition for $\frac{1}{100}$ reduction of learning set.

	Original Set	Lloyd	Neural Gas	k-means
Trained Patterns	100%	60.3%	56.4%	48.9%
Speaker Dependen.	62.7%	50.6%	49.2%	38.2%
Speaker Independ.	39.6%	36.1%	38.8%	41.1%
Mean Results	67.4%	49.0%	48.1%	42.7%

4.2 Word Recognition Results

The complete word recognition was performed using the previous estimated phone chains as input for a DTW system. The accuracy levels reached with the clustering methods were comparable among the proposed methods but with a slight advantage when using the LBG method for high reductions ($\frac{1}{100}$). The complete word recognition results are outline in table 5. These results show the mean recognition rates of all the different learning sets under the speaker independent tests.

Table 5. Speaker independent word recognition results for all methods.

	Original Set	$\frac{1}{10}$ Reduc.	$\frac{1}{100}$ Reduc.
LBG method	96%	91%	89%
Neural Gas	96%	91%	88%
K-means	96%	92%	84%

5 Concluding Remarks

Three competitive learning methods, the *LBG*, *Neural Gas* and *K-means* algorithms, were successfully used for reducing the dimension of learning files without significantly loosing the accuracy level for speech recognition tests. The three algorithms divide the learning space into no overlapping regions, where each region is bounded by a number of reduced hyper-planes. Each region is characterized

by the occurrence of a reduced number of the total alphabet of phones (basic speech units). The classification of test patterns is performed by identifying the region in which the pattern lies, remaining the learning vectors that exist in that particular region, the best attractors for the phone estimation. The use of the proposed competitive learning methods, permitted to operate the speech recognizer in an optimized and computationally inexpensive way.

The three studied methods show in general, to have a comparable performance for the particular application of speech recognition. In this way, the contribution of this paper is related to the possibility of selecting any one of the proposed methods for efficient data selection. The previous is a very important task in speech recognition problems for optimizing in time and computer resources. Nevertheless the similarity of all the proposed methods, some slightly different results were observed for the two proposed levels of reduction ($\frac{1}{10}$ and $\frac{1}{100}$). In the first case, the k-means method show a better performance over the Lloyd's and Neural gas techniques while for the second level of reduction the performance was inverse. Additionally, the batch methods demonstrated to work better for the speaker dependent tests while the on-line K-means technique obtained better results for different speakers. In every case, the selection of one of the presented methods rather than any other one, would not depend in the obtained accuracy levels (that show to be very similar) but in the specific requirements of the particular application, such as the computational complexity and the time constraints.

In the specific studied case, the reduction level of near of two orders of magnitude of the original learning set, permitted to strongly reduce the time and computational resources needed for training and testing the parallel associative memory for speech recognition.

References

1. M. Mohri and M. Riley, "Weighted Determination and Minimization for Large Vocabulary Speech Recognition", *Proc. of Eurospeech'97*, Vol. 1, pp. 131-134, Rhodes, Greece, 1997.
2. M. Padmanabhan, L. R. Bahl, D. Nahamoo, and P. de Souza, "Decision-Tree Based Quantization of the Feature Space of a Speech Recognizer", *Proc. of Eurospeech'97*, Vol. 1, pp. 147-150, Rhodes, Greece, 1997.
3. M. Ravishankar, R. Bisiani and E. Thayer, "Sub-Vector Clustering to Improve Memory and Speed Performance of Acoustic Likelihood Computation", *Proc. of Eurospeech'97*, Vol. 1, pp. 151-154, Rhodes, Greece, 1997.
4. K. Paliwal, B. Atal, "Efficient Vector Quantization of LPC Parameters at 24 Bits/Frame", *IEEE Transactions on Speech and Audio Processing*, Vol. 1, No. 1, Jan. 1993, pp. 3-14.
5. Y. Linde, A. Buzo and R.M. Gray, "An algorithm for vector quantizer design", *IEEE Transactions on Communication*, COM-28, pp. 84-95, 1980.
6. T.M. Martinetz and K.J. Schulten., "A Neural Gas Network Learns Topologies", In T. Kohonen, K. Maklsara, O. Simula and J. Kangas editors, *Artificial Neural Networks*, pp. 397-402, North Holland, Amsterdam,1991.

7. J. Mac Queen, "Some methods for classification and analysis of multivariate observations", *Proc. of the Fifth Berkeley Symposium on Mathematical statistics and probability*, pp. 281-297, Berkeley, 1967.
8. M.G. Voronoi, "Nouvelles applications des parametres continus a la theorie des formes quadratiques", *J. Reine u. Angew. Math.*, 134:198-287, 1908.
9. V. Cherkassky and F. Mulier, "Learning from data: Concepts, theory and methods", *John Wiley and Sons*
10. B. Fritzke, "Some Competitive Learning Methods", *Technical Report, Institute for Neural Computation*, Ruhr-Universitat Bochum, 1997.
11. H. Hermansky, "Perceptual Linear Predictive (PLP) Analysis of Speech", *Journal of Acoust. Soc. Am.*, pp. 1738-1752, April 1990.
12. A. Robinson, "SHORTEN: Simple lossless and near-lossless waveform compression", *Technical Report, CUED/F-INFENG/TR 156* Cambridge University, UK., Dec. 1994.
13. SPK Isolated Digit Database. ELRA-IRST. Istituto per la Ricerca Scientifica e Tecnologica.
14. A. Robinson, "An Application of Recurrent Nets to Phone Probability Estimation", *IEEE Transactions on Neural Networks*, Vol. 5, No. 2, March 1994, pp. 298-305.
15. F. Curatelli, O. Mayora-Ibarra, D. Carotenuto, "SPEAR, A Modular Tool for Speech Signal Processing and Recognition", *Proceeding of WISP'99*, Budapest, Hungary, 1999.
16. F. Curatelli, O. Mayora-Ibarra, "An Hybrid Parallel Associative Memory / DTW Based System for Speech Recognition", *In Recent Advances in Signal Processing and Communications*, World Scientific Engineering Society, pp. 140-144, 1999.

Facial Expression Recognition and Modeling for Virtual Intelligent Tutoring Systems

Homero V. Ríos[1], Ana Luisa Solís[2], Emilio Aguirre[1], Lourdes Guerrero[2], Joaquín Peña[1], and Alberto Santamaría[1]

[1] Laboratorio Nacional de Informática Avanzada, A.C.
Rébsamen 80, c.p. 91090, Xalapa, Veracruz, México
[2] Departamento de Matemáticas, Facultad de Ciencias,
Universidad Nacional Autónoma de México

Abstract. This paper describes ongoing work for developing a new interface for intelligent tutoring systems based on recognition and synthesis of facial expressions. This interface senses the emotional state of the user, or his/her degree of attention, and communicates more naturally through face animation.

1 Introduction

This work describes the ongoing development of a new interaction technique for intelligent tutoring systems based on the integration of results from computer graphics and computer vision. In the last few years this integration has shown important results and applications [7,15]. For example, Richard Szeliski described the use of image mosaics for virtual environments in 1996 and, in the following year for combining multiple images into a single panoramic image. H. Ohzu et al. described hyper-realistic communications for computer supported cooperative work [12].

Facial expression understanding is a good example of the rich middle ground between graphics and vision. Computer vision provides an excellent input device, particularly for the shapes and motions of complex changing shapes of faces when expressing emotions [15,14].

We have been studying how to analyze video sequences for capturing gestures and emotions. Relevant expressions and their interpretations may indeed vary depending upon the chosen type of application [17].

In this work, relevant expressions are assigned with a tutorial system when it is important to know the user interest on the information that is displayed or when she/he is interacting in a virtual environment.

Facial expression recognition is useful for adapting interactive feedback in a tutoring system based on the students level of interest. The type of expressions associated with these applications are: degree of interest, degree of doubt of the information presented, boredom among other expression, or to assess the time of interest or lack of interest presented by an application.

The work mentioned here also strives to capture the high resolution motion and appearance of an individual face. The goal is to use this information

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 115–126, 2000.

to animate and render synthetic faces and to have interaction with a tutorial system.

2 Analysis and Interpretation of Facial Expressions

The communicative power of faces makes it a focus of attention during social interaction. Facial expressions and the related changes in facial patterns inform us of the emotional state of people and help to regulate both social interactions and spoken conversation. To fully understand the subtlety and expressive power of the face, considering the complexity of the movements involved, one must study face perception and related information processing.

For this reason, face perception and face processing have become major topics of research by cognitive scientists, sociologists and more recently by researchers in computer vision and computer graphics.

The automation of human face processing by computer will be a significant step towards developing an effective human-machine interface. We must consider the ways in which systems with this ability understand facial gestures (analysis), and the means of automating this interpretation and/or production (synthesis) to enhance human-computer interaction.

2.1 Facial Displays as a New Modality in Human-Computer Interaction

Facial expressions can be viewed in either of two ways. One regards facial gestures as expressions of emotional states. The other view is facial expressions related to communication. The term "facial displays" is equivalent to "facial expressions", but does not have the connotation of emotion.

The present paper assumes the second more general view. A face as an independent communication channel. A facial display can be also seen as a modality because the human brain has special neurons dedicated to the necessary processing.

Part of this research is an attempt to computationally capture the communicative power of the human faces and to apply it to user interfaces.

2.2 Theory of Communicative Facial Displays

First of all, facial displays are primarily communicative. They are used to convey information to other people. The information that is conveyed may be emotional, or other kinds of information, indications that the speaker is being understood, listener responses, etc.

Facial displays can function in an interaction as means of communication on their own. That is, they can send a message independently of other communicative behavior. Facial signs such as winks, facial shrugs, and listener's comments (agreement or disagreement, disbelief or surprise) are typical examples. Facial displays can also work in conjunction with other communicative behavior (both verbal and nonverbal) to provide information.

2.3 Communicative Categorization of Facial Displays

Some of the facial displays that we are trying to incorporate for tutoring systems are the followings:

- *Thinking/Remembering.* Eyebrow raising or lowering. Closing the eyes. Pulling back one mouth side.
- *Facial shrug/I don't know.* Eyebrow flashes. Mouth corners pulled down. Mouth corners pulled back.
- *Backchannel/Indication of attention.* Eyebrows raising. Mouth corners turned down.
- *Doubt.* Eyebrows drawn to center.
- *Undertanding levels.*
  - *Confident.* Eyebrows raising, Head nod.
  - *Moderately confident.* Eyebrows raising.
  - *Not confident.* Eyebrows lowering.
  - *"Yes".* Eyebrows raising.
- *Evaluation of utterance.*
  - *Agreement.* Eyebrows raising.
  - *Request for more information.* Eyebrows raising.
  - *Incredulity.* Longer eyebrows raising.

2.4 Face Analysis

The analysis of faces by computer is difficult since there are several factors that influence the shape and appearance of faces on images. Some of these factors are illumination, viewpoint, color, facial hair and the variability of faces. In addition, we still do not know the exact mechanisms used by humans for face processing. For instance is face processing a holistic or feature analysis process?. The brain itself has specialized structures like IT cells on the visual areas to handle face detection [1].

The problem of face analysis can be divided in face detection and face recognition. In the first case the goal is just to locate the general position of faces on images. On the latter, the purpose is to recognize faces using extracted features. This recognition can take place on the following conditions [3]: a) static images, b) range images, and c) video sequences.

Since faces are non-rigid objects the best way to model them is through the use of deformable models which should have enough parameters to accommodate most of the variations in shape. Active contours have been used to detect and track facial features like head contour, lips, eyebrows and eyes [10,18]. The problem is that since "snakes" can adapt any shape, sometimes they take nonvalid shapes. One solution is the use of trainable deformable models from examples, like the point distribution model proposed by Cootes [5]. In our case, we have used this approach to detect facial features and interpret facial gestures. The difference between this work and ours, is that we have built a database of faces which shows more clearly different gestures. Also, our model has been expanded to include landmarks which enhances the identification of facial gestures.

To train the deformable model for face detection we have used a database containing 145 faces (Figure 1). Some of the images are from the public domain database: http://peipa.essex.ac.uk/ipa/pix/faces/manchester/train, and the rest are from our database, built to show different gestures.

Each face is represented by vector $\mathbf{x} = (x1, y1, ..., x_N, y_N)$ with 2N entries, where N is the number of landmarks used to approximate the face shape. In our case, we have used 127 landmarks per face (Figure 2 and 3).

Fig. 1. Some of the images on the training set.

After aligning the training faces to reduce the effect of translation, rotation and scaling, principal component analysis is used to obtain the main modes of variation [5]. 33 eigenvectors account for 95% of shape variation (Figure 4 and 5). Any shape $\mathbf{x_i}$ in the training set can be approximated using the mean shape $\overline{\mathbf{x}}$ and a weighted sum of the first t eigenvectors (ordered from the most significant eigenvalue to the least significant) as follows:

$$\mathbf{x_i} = \overline{\mathbf{x}} + \mathbf{Pb} \tag{1}$$

where $\mathbf{P} = (\mathbf{p_1}, ..., \mathbf{p_t})$ is the matrix of first t eigenvectors of the covariance of deviations, and $\mathbf{b} = (b_1, ..., b_t)$ is a vector of t eigenvector weights. By modifying $\mathbf{b}$ within some limits, new instances of the model can be generated. The limits are typically $[-3\sigma_k, +3\sigma_k]$, where σ_k is the standard deviation of $\mathbf{b}_k$ over the training set.

One importance difference between our work and the research done by Cootes et al. [5], is that our database and face model has been built specifically to extract and identify facial gestures. Since our database has more individuals and gestures, we need more eigenvectors to represent the variability in the data set. Also, more that one eigenvector is needed to represent the "universal" gestures (fear, anger, surprise, disgust, happiness, and sadness).

The deformable model is fitted using an active shape model as proposed by Cootes et al. [5]. Examples of the model fit are shown on figures 6-9. From the extracted representation we proceed to expression recognition and face animation.

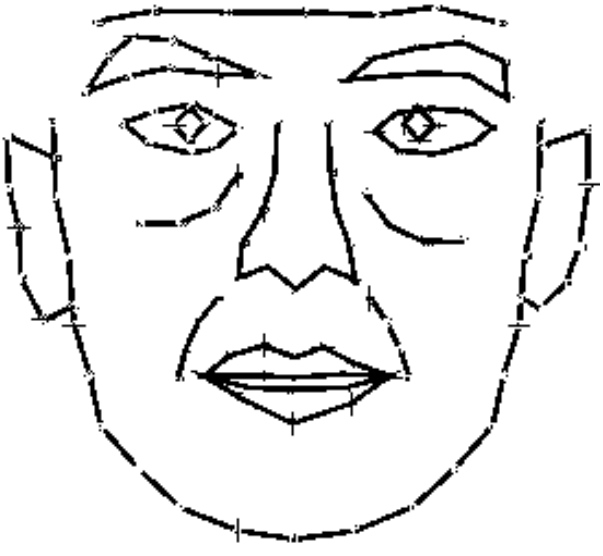


Fig. 2. Landmarks used for face representation.

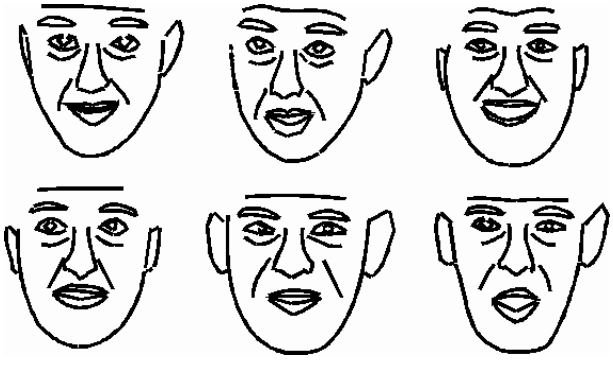


Fig. 3. Some of the contours extracted from the training set.

3 Recognition and Modeling of Facial Expressions for Tutorial Systems

First, we must considered the facial expressions we are interested in recognize related with Tutorial System. This kind of expressions are very specific: doubt with different levels of intensity, understand, no understand, etc.

To categorize expressions, we need first to determine the expressions from facial movements. Ekman and Friesen have produced a system describing all visually distinguishable facial movements. The system, called the Facial Action Coding System or FACS, is based on the enumeration of all "Action units" of a face that cause facial movements. As some muscles give rise to more than

Fig. 4. Mean shape.

one action unit, the correspondence between action units and muscle units is approximate.

3.1 Facial Action Coding System

The Facial Action Coding System (FACS) describes the set of all possible basic action unit (AUs) performable by the human face. Each action unit is a minimal action that cannot be divided into smaller actions. According to Ekman, "FACS allows the description of all facial behavior we have observed, and every facial action we have attempted".

The primary goal of the FACS was to develop a comprehensive system which could reliably describe all possible visually distinguishable facial movements.

The use of FACS in facial animation goes beyond what was originally intended. In a number of facial animation systems, FACS is used as a way to control facial movement by specifying the muscle actions needed to achieve desired expression changes.

FACS was derived by analysis of the anatomical basis for facial movements. Since every facial movements is the result of some muscular action, the FACS system was developed by determining how each muscle of the face acts to change appearance.

Ekman's FACS model has 46 major Action Units and divides the face into three areas: brows, eyes and lower face. The association between these action units and the muscles on the face can be checked on reference [8,9].

3.2 Synthesis and Animation of Expression

Facial animation typically involves execution of a sequence of a set of basic facial actions. We use action units (AU) of the Facial Action Coding System (FACS) as atomic action units and as basic facial motion parameters (Figure 10).

Each AU has a corresponding set of visible movements of different parts of the face resulting from muscle contraction. Muscular activity is simulated using

Fig. 5. 33 eigenvectors account for 95% of shape deformation. Here we show the most significant for gesture recognition. Eigenvectors: 1-5, 8, 9, 12. Each row represents the mean face plus an eigenvector multiplied by a weight factor. From left to right the weight factor varies in the range $[-3\sigma_k, +3\sigma_k]$, where σ_k is the square root of the i'th eigenvalue.

a parameterized facial muscle process. We can define atomic action units similar to AU and definite expressions and phonemes. These can be used for defining emotions and sentences for speech.

For the driven facial animation the input parameters to the facial animation are the AUs. The facial expression recognition module provides a two way dialog and requires that the person on the other side to have a camera (Figure 11).

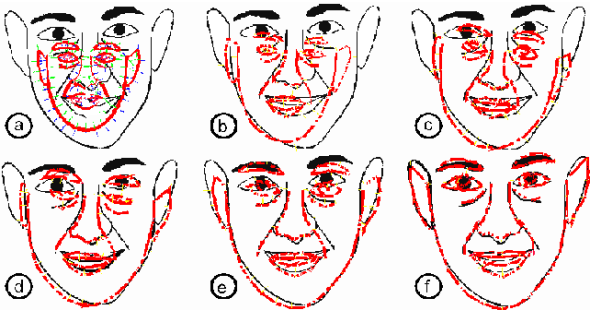


Fig. 6. Model fitting on binary images.

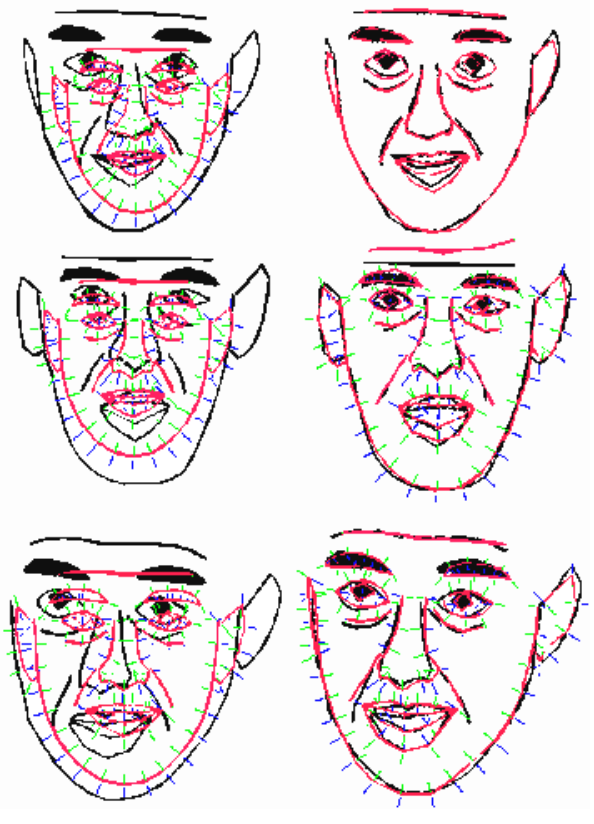


Fig. 7. Another example of model fitting on binary images.

This module would perform much more task than just merely copying other's facial expression. We are developing a prototype for a virtual dialog system where a virtual actor communicates with a real person by the analysis of facial expression.

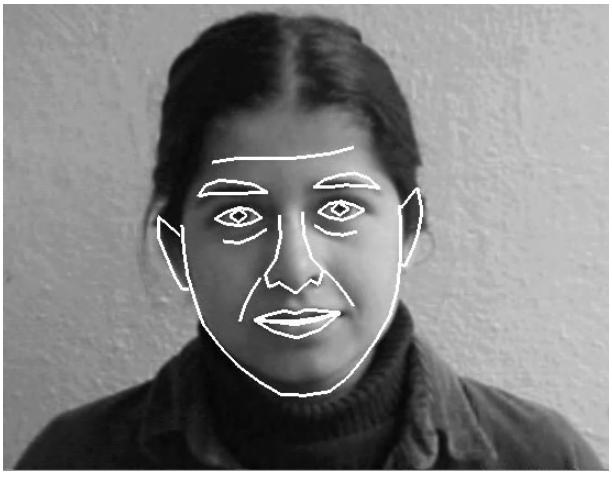


Fig. 8. Model fitting on grayscale images.

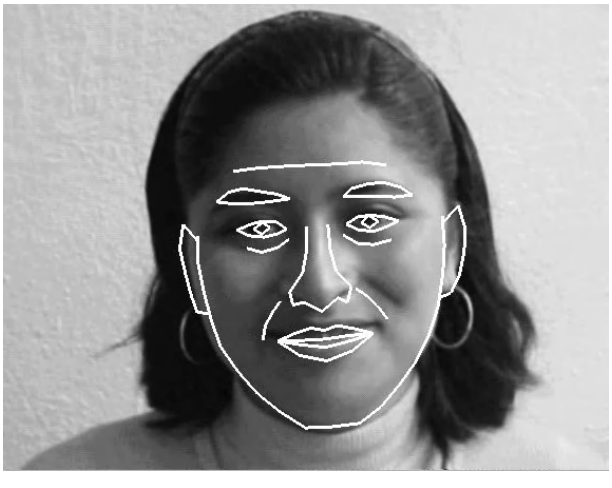


Fig. 9. Model fitting on grayscale images.

4 Integration of a Intelligent Tutoring Systems (ITS) in a Virtual Environment

4.1 Face Expression Input for the ITS

The system is well suited for developing an autonomous intelligent tutor who may communicate and exchange a dialog with a real person through expression recognition.

We want to propose a simple architecture (Figure 12). The input to the analyzer is the facial expression recognition module. The result of the analysis

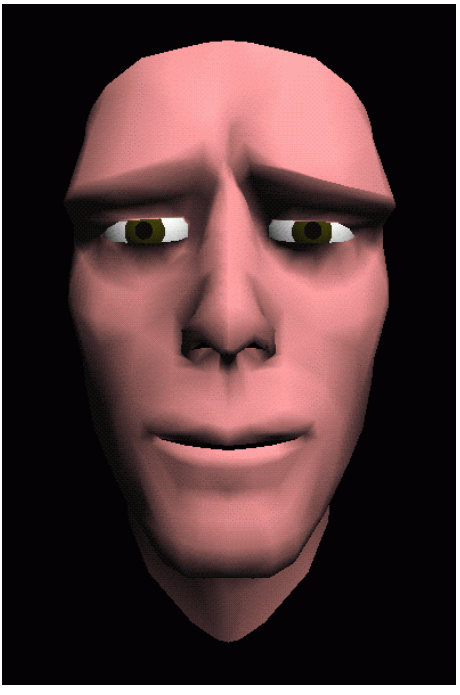


Fig. 10. A facial expression of doubt.

Fig. 11. Facial communication.

can provide the expression recognition of the user. The virtual actor responds to a real person, a database is used with content words with facial expression states. These are defined in terms of constituent phonemes and facial expressions.

The series of timed phonemes and expressions are then compiled and decomposed into the basic action units to produce the visual changes to be done on the virtual tutor's face. The words to be spoken are transmitted to the vocal synthesizer module.

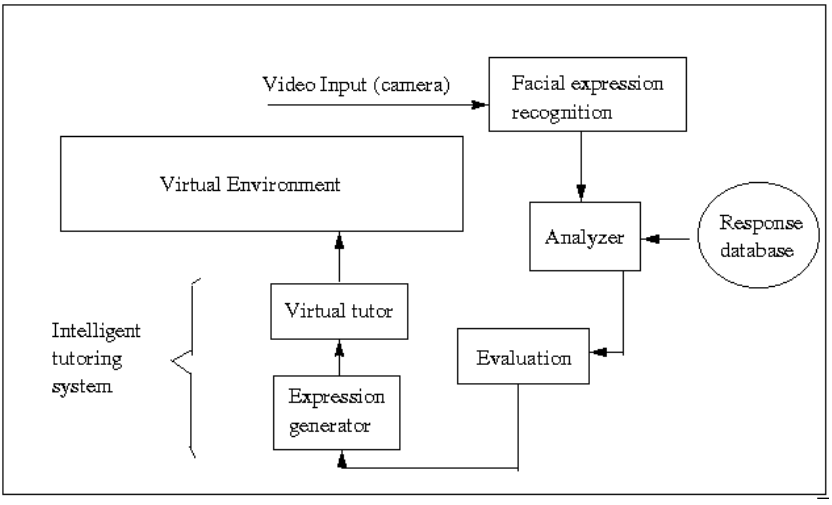


Fig. 12. Global arquitecture of the system.

4.2 Interacting with the ITS Through Face Expression

Input from facial expression may be used to interact with an application in a virtual environment. Relevant expressions are assigned with a virtual tutorial system when it is important to know the user interest on the information that is displayed or when the user is interacting in a virtual environment.

5 Conclusion

In this paper we have described our ongoing work on a human-computer interface based on face analysis and synthesis, that enhances the communicative power of intelligent tutoring systems. The analysis allows facial expression recognition, while synthesis renders a realistic virtual tutor which could be complemented with synthetic speech.

Acknowledgements. This work has been funded by the Mexican National Council of Science and Technology (CONACYT) as project ref. C098-A and C100-A, "Gesture recognition interfaces and intelligent agents for virtual environments".

References

1. Bruce, V., Green, P.: Visual Perception: Physiology, Psychology and Ecology. Lawrence Erlbaum Associates, London. (1989)
2. Cassell, J., Pelachaud, C. Badler, N. et al.: Animated Conversation: Rule-based Generation of Facial Expression, Gesture and Spoken Intonation for Multiple Conversational Agents. Computer Graphics Proceedings (1994) 413–420

3. Chellapa, R., Wilson, C.L., Sirohey, S.: Human and machine recognition of faces: a survey. Proceedings of the IEEE, Vol. **83** (1995)705–740
4. Cohen, M. and Massaro, D.: Modeling Coarticulation in Synthetic Visual Speech, Models and Techniques in Computer Animation. Springer Verlag (1993) 139–156
5. Cootes, T.F., Taylor, C.J., Cooper, D.H., Graham, J.: Active shape models–Their training and applications. Computer Vision, Graphics and Image Understanding **61** (1995) 38-59
6. Cootes, T.F., Edwards, G.J., Taylor, C.J.: Active Appearance Models. In: Burkhardt, H., Neumann, B. (eds.): Proceedings of the European Conference on Computer Vision. Springer-Verlag (1998) 484–498
7. Eisert, P., Girod, B.: Analyzing facial expressions for virtual conferencing. IEEE Computer Graphics and Applications, **18** (1998) 70–78
8. Ekman, P., Friesen, W.V.: Unmasking the Face: A Guide to Recognizing Emotions from Facial Expressions. Prentice Hall, Inc. (1975)
9. Ekman, P., Friesen, W.V.: Manual for the Facial Action Coding System. Consulting Psychologists Press, Inc. Palo Alto, CA. (1978)
10. Lam, K.M. and Yang, H.: Locating and extracting the eye in human face images. Pattern Recognition, **29**(1996) 771–779
11. Lanitis, A., Taylor, C.J., Cootes, T.F.: Automatic interpretation and coding of face images using flexible models. IEEE Transactions on Pattern Analysis and Machine Intelligence **19** 743-756
12. Ohzu, H., Habara, K.: Behind the scenes of virtual reality: vision and motion. Proceedings of the IEEE, **84** (1996) 782–798.
13. Maggioni, C., Kammerer, B.: GestureComputer - History, Design and Applications. In: Cipolla, R., Pentland, A. (eds.): Computer Vision for Human-Machine Interaction. Cambridge University Press (1998) 23–52
14. Parke, F.I., Waters, K.: Computer Facial Animation. A K Peters (1996)
15. Pentland, A.P.: Smart rooms. Scientific American. April (1996) 54–62.
16. Pentland, A.P.: Smart rooms: Machine understanding of human behavior. In:Cipolla, R., Pentland, A. (eds.): Computer Vision for Human-Machine Interaction. Cambridge University Press (1998) 3–22
17. Ríos, H.V. and Peña, J.: Computer Vision interaction for Virtual Reality, In: Coelho, H. (ed.): Progress in Artificial Intelligence. Lecture Notes in Artificial Intelligence, **1484**. Springer-Verlag, (1998) 262–273
18. Terzopoulos, D., Szeliski, R.: Tracking with Kalman Snakes. In: Blake, A., Yuille, A. (eds.): Active Vision. MIT Press(1992) 3–20
19. Waters, K.: A Muscle Model for Animating Three Dimensional Facial Expression. Computer Graphics Proceedings **21**(1987) 17–23.
20. Waters, K., Levergood, T.: An Automatic Lip-Synchronization Algorithm for Synthetic Faces. Proceedings of Multimedia 94, ACM. (1994) 149–156.

Detection and Tracking of Facial Features in Video Sequences

Rogério Schmidt Feris, Teófilo Emídio de Campos, and Roberto Marcondes Cesar Junior

CreatiVision - Creative Vision Research Group
Department of Computer Science
DCC-IME-USP, University of São Paulo
Rua do Matão, 1010, São Paulo, SP, 05508-900, Brazil
Phone: +55 (011) 818 6235, Fax: +55 (011) 818 6134
{rferis, teo, cesar} @ime.usp.br

Abstract. This work presents a real time system for detection and tracking of facial features in video sequences. Such system may be used in visual communication applications, such as teleconferencing, virtual reality, intelligent interfaces, human-machine interaction, surveillance, etc. We have used a statistical skin-color model to segment face-candidate regions in the image. The presence or absence of a face in each region is verified by means of an eye detector, based on an efficient template matching scheme . Once a face is detected, the pupils, nostrils and lip corners are located and these facial features are tracked in the image sequence, performing real time processing.

1 Introduction

The detection and tracking of faces and facial features in video sequences is a fundamental and challenging problem in computer vision. This research area has many applications in face identification systems, model-based coding, gaze detection, human-computer interaction, teleconferencing, etc.

In this paper, we present a real time system that performs facial features detection and tracking in image sequences. We are interested in developing intelligent interfaces and this system is the first step to give computers the ability to look at people, which may be a good way to improve human-computer interaction.

We have used a color-based approach to detect the presence of a human face in an image. A Gaussian distribution model was generated by means of a supervised training, using a set of skin-color regions, obtained from a color face database.

Basically, the input image is compared with the generated model in order to identify face candidates, i.e., skin-color regions. The presence or absence of a face in each region is verified by using an eye detector based on an efficient template matching scheme. There is biological evidence that eyes play the most important role in human face detection [1].

Once a face is detected, the pupils, nostrils and lip corners are located and tracked in the video sequence. The system is very fast and robust to small face pose and rotation variation.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 127—135, 2000.

The remainder of this paper is organized as follows. Section two reviews some techniques related to our work. Section three describes the statistical skin-color model and section four explains how it was used to segment face-candidate regions. Section five describes the eye detector used to perform the face verification process. In section six, the detection and tracking of facial features is described. The obtained results are showed in section seven. Finally, in section eight follows a conclusion and we address further research directions.

2 Related Work

Automatic detection of facial features in video sequences is performed, in general, after a face is detected in a frame. Many approaches have been proposed to detect faces in static scenes, such as Principal Component Analysis [1,2], clustering [3] and neural nets [4]. These techniques obtained good detection / false alarm rates, but they are computational expensive and hardly achieve real-time performance.

Liyanage Silva et. al. [5] proposed a method, which they called edge pixel counting, to detect and track facial features in video sequences. This method is based on the fact that a higher edge concentration is observed in the vicinity of facial features, while slightly outside of such features a less edge concentration is observed. In spite of its simplicity, the method fails in the presence of cluttered backgrounds, glasses, hat, etc.

Recently, several color-based systems have been proposed to perform face detection and tracking in image sequences. Processing color is much faster than processing other facial features and, furthermore, color is orientation invariant under certain lighting conditions. This property makes motion estimation much easier since only a translation model is needed.

The work of Jie Yang and Alex Waibel [6] presents a statistical skin-color model, adaptable to different people and different lighting conditions. This work was extended to detect and track faces in real time [7], achieving a rate of 30+frames/second using a HP-9000 workstation with a framegrabber and a Canon VC-C1 camera.

Nuria Oliver et. al. [8] and Trevor Darrell et. al. [9] also obtained good results using skin-color to detect and track people in real time. Stiefelhagen and Yang [10] presented a method similar to ours, which detects and tracks facial features in image sequences. The main differences lie in face verification process. They have used iterative thresholding to search the pupils whereas we have adopted an efficient template matching scheme to detect eyes.

3 Skin-Color Model

The statistical skin-color model [6] is generated by means of a supervised training, using a set of skin-color regions, obtained from a color face database, with 40 face images. Such images were obtained from people of different races, ages and gender, with varying illumination conditions. Figure 1 illustrates the training process, in which

a skin-color region is selected and its RGB representation is stored. It was verified, using training data, that skin-colors are clustered in color space, as illustrated in Figure 2.

It is common to think that different people have skin-colors which differ significantly from each other due to the existence of different races. However, what really occurs is a larger difference in brightness / intensity and not in color [6]. Thus, it is possible to reduce the variance of the skin-color cluster through intensity normalization:

$$a = \frac{R}{R+G+B} \tag{1}$$

$$b = \frac{G}{R+G+B} \tag{2}$$

The colors (a,b) are known as chromatic or "pure" colors. Figure 3 illustrates the skin-color cluster in chromatic space.

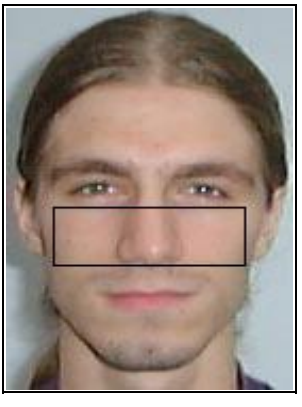


Fig. 1. Selected skin-color region

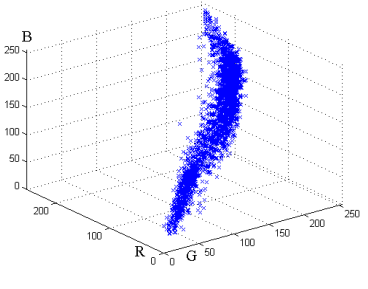


Fig. 2. Cluster in color space

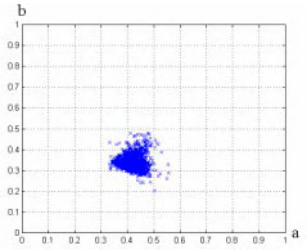


Fig. 3. Cluster in chromatic space

According to [6], the skin-color distribution in chromatic space can be approximated by a Gaussian model $N(m,\Sigma^2)$, where $m=(a_{avg},b_{avg})$ and:

$$a_{avg} = \frac{1}{N}\sum_{i=1}^{N} a_i \tag{3}$$

$$b_{avg} = \frac{1}{N}\sum_{i=1}^{N} b_i \tag{4}$$

$$\Sigma = \begin{bmatrix} \sigma_{aa} & \sigma_{ab} \\ \sigma_{ba} & \sigma_{bb} \end{bmatrix} \tag{5}$$

The Gaussian model, which was generated from training data, is illustrated in Figure 4.

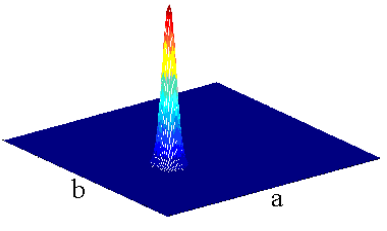


Fig. 4. Gaussian Model

4 Image Segmentation

The segmentation process consists in identifying skin-color regions in the input image (the first frame of the video sequence) based on the generated model.

Initially, all pixels in input image are converted to the chromatic space. Using the Gaussian model, a gray-level image is generated and the intensity of each pixel in this

new image represents its probability to have skin-color. Figure 5 illustrates this process.

Since skin-color pixels present a stronger response in the gray-level image, we can identify such regions by a thresholding process. The threshold value was chosen empirically.

In general, due to noise and distortions in input image, the result of thresholding may generate non-contiguous skin-color regions. Furthermore, after this operation, spurious pixels may appear in the binary image. In order to solve these problems, first we have applied a morphological closing operator to obtain skin-color blobs. Subsequently, a median filter was used to eliminate spurious pixels. Figure 6 shows the obtained binary image after thresholding (left illustration) and the result of morphological closing operation followed by a median filtering (right illustration).

Fig. 5. Skin-color identification

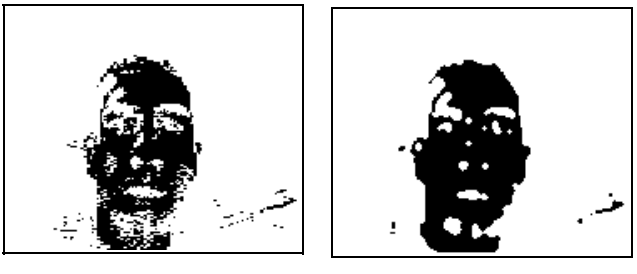


Fig. 6. Thresholding and morphological closing followed by median filtering

Boundaries of skin-color regions are determined using a region growing algorithm in the binary image. Regions with size less than 1% of image size are eliminated. Furthermore, structural aspects are considered so that regions that do not present the face structure are removed.

In the next section, we will show a method to decide whether each detected skin-color region corresponds or not to a face.

5 Verification Process

The main goal in identifying skin-color regions in the input image is to reduce the search space for faces. However, it is important to note that not all detected regions contain faces. Some correspond to parts of human body, while other correspond to

objects with colors similar to those of skin. Thus, it is necessary to verify the presence or absence of a face in each detected region.

To accomplish this task, we have used an eye detector based on an efficient template matching scheme. It is worth saying that there is biological evidence that eyes play the most important role in human face detection [1]. The eye template used is illustrated in Figure 7. Its resolution is 57x15 with 256 gray levels.

Fig. 7. Eye template

The eye detector procedure first converts the skin-color region to a gray-level image, in order to perform template matching. The most important property in our technique is that the eye template is resized according to the skin region size. In general, existing methods use several templates with different scales to accomplish the detection. For instance, the work of Brunnelli and Poggio [11] uses five templates with scales 0.7, 0.85, 1, 1.5 and 1.35 to detect eyes in an image. Although these methods have obtained good results, they are computational expensive and not suitable to real time processing.

Our method estimates the person eye size, resizing the eye template. Thus, the template matching is performed only once, using the upper portion of the detected skin-color region. Basically, correlation coefficients ρ are computed in different positions within the skin region:

$$\rho = \frac{\mathrm{cov}(x, y)}{\sigma(x)\sigma(y)} \tag{6}$$

where x is the eye template and y is the image patch in a specific position within the skin region. The function *cov* in the equation above returns the covariance between x and y. The correlation coefficient ρ assume values ranging from -1 to 1.

A threshold value, which was determined empirically, was used to decide eye detection. When eyes are found within the skin-color region, we report the presence of a face. The system then detects some facial features and proceeds in the tracking module, as described in the next section.

It is important to note that the above strategy detects the first frame in which the face appears in frontal view.

6 Detection and Tracking of Facial Features

Once a face is detected, the system proceeds with the search for pupils, nostrils and lip corners. After, these facial features are tracked in the video sequence. The approach used is similar to the work of Stiefelhagen and Yang [10].

6.1 Searching the Pupils

Assuming a frontal view of the face initially, we locate the pupils by looking for two dark pixels that satisfy certain geometric constraints and lie within the eye region detected by template matching.

6.2 Searching the Lip Corners

First, the approximate positions of lip corners are predicted, using the position of the face and the pupils. A small region around those points is then extracted and a Sobel horizontal edge detector is applied. The approximate horizontal boundaries of the lips are determined by first computing the vertical integral projection P_v of this horizontal edge image:

$$P_V(y) = \sum_{x=1}^{H} E_h(x, y), \quad 0 \le y \le W \tag{7}$$

where $E_h(x,y)$ is the intensity function of the horizontal edge image, and W and H are the width and height of the search region, respectively.

The approximate left and right boundaries of the lips are located where P_v exceeds a certain threshold t or respectively falls below that threshold. We choose t to be the average of the projection P_v. The vertical position of the left and right lip corners are simply located by searching for the darkest pixel along the columns at the left and right estimated boundaries of the lips in the search region.

6.3 Searching the Nostrils

Similar to searching the pupils, the nostrils are located by searching for two dark regions, which satisfy certain geometric constraints. The search region is restricted to an area below the pupils and above the lips.

Figure 7 illustrates the detection of facial features.

Fig. 7. Detection of facial features

6.4 Tracking

For tracking pupils and nostrils in the video sequence, simple darkest pixel finding in the search windows around the last positions is used. In order to track lip corners, the same detection process is performed in a small region around the last lip corners position.

7 Obtained Results

The proposed real time facial feature tracker obtained good results in different video sequences. Initially, a color-based frontal view face detection is performed. This procedure fails, in general, when the eye template scale is incorrectly estimated. This case occurs when the skin region contains, beyond a face, other human body parts, such as a hand occluding the face, or other objects. We have also noted that faces may be missed when they are too small or under strong varying illumination.

The face detection procedure has proven to be robust in identifying skin-colors of people with different races, ages and gender. It is worth saying that dark-skin regions had smaller skin-color probabilities, since most persons in the face database had light skin.

Once a face is detected, its facial features are located and tracked in the image sequence. If there is more than one face in the first frame, only the bigger face is considered.

During the tracking process, we verified that the system fails when a large movement is performed by the person. Furthermore, due to varying illumination, some features may be missed in the sequence. On the other hand, the tracking module showed to be robust to small face pose and rotation variations.

Figure 8 shows the tracking process in some frames of a image sequence, which can be observed in *http://www.ime.usp.br/~cesar/creativision*.

Fig. 8. Tracking Process

8 Conclusions

This paper described a real time system for detection and tracking of facial features in video sequences. A statistical skin-color model was used to detect face-candidate regions and an efficient template matching scheme was applied to report the presence or absence of a face in these regions. Given a detected face, pupils, nostrils and lip corners are located and tracked in the image sequence. The system may be used in different applications, such as teleconferencing and human-computer interaction.

Future work includes pose estimation and 3D head tracking by using the positions of pupils, nostrils and lip corners. Furthermore, we intend to improve the system robustness developing a recovery module and considering motion to increase the accuracy of detection and tracking of facial features.

Acknowledgments. Roberto M. Cesar Jr. is grateful to FAPESP for the financial support (98/07722-0), as well as to CNPq (300722/98-2). Rogério Feris and Teófilo Campos are grateful to FAPESP (99/01487-1 and 99/01488-8).

We are grateful to Carlos Hitoshi Morimoto (IME-USP) for useful discussions.

References

[1] B. Moghaddam and A. Pentland. “Face Recognition using View-Based and Modular Eigenspaces”. Proc. of Automatic Systems for the Identification and Inspection of Humans, SPIE vol. 2277, July 1994.

[2] B. Moghaddam and A. Pentland. “Probabilistic Visual Learning for Object Detection”. In Fifth International Conference on Computer Vision, pp. 786-793, Cambridge, Massachusetts, June 1995

[3] K. Sung and T. Poggio. “Example-base Learning for View-based Human Face Detection” C.B.C.L. Paper no. 112, MIT, 1994.

[4] H. Rowley, S. Baluja and T. Kanade. “Neural Network-based Face Detection”. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 1, pp. 23-38, January 1998.

[5] L. Silva, K. Aizawa and M. Hatori. “Detection and Tracking of Facial Features”. Proc. of SPIE Visual Communications and Image Processing, Taiwan. May, 1995.

[6] J. Yang, W. Lu and A. Waibel. “Skin-Color Modeling and Adaptation”. CMU CS Technical Report, CMU-CS-97-146. May, 1997.

[7] J. Yang and A. Waibel. “A Real-time Face Tracker”. Proc. of the Third IEEE Workshop on Applications of Computer Vision, pp. 142-147, Sarasota, Florida, 1996.

[8] N. Oliver, A. Pentland and F. Berard. “LAFTER: Lips and Face Real-time Tracker with Facial Expression Recognition”. Proc. of Computer Vision and Pattern Recognition Conference, 1997.

[9] C. Wren, A. Azarbayejani, T. Darrell and A. Pentland. “Pfinder: Real-time Tracking of the Human Body”. Proc. SPIE, vol. 2615, pp. 89-98, 1996.

[10] R. Stiefelhagen and J. Yang. “Gaze Tracking for Multimodal Human Computer Interaction”. University of Karlsruhe, 1996. Available at http://werner.ira.uka.de/ISL.multimodal.publications.html.

[11] R. Brunelli and T. Poggio. “Face Recognition: Features versus Templates”. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, no. 10, pp. 1042-1052. October 1993.

An Application of Behavior-Based Architecture for Mobile Robots Design

Sergio Uribe-Gutierrez and Horacio Martinez-Alfaro

Centro de Inteligencia Artificial
ITESM Campus Monterrey
Ave. Eugenio Garza Sada 2501
Monterrey, N.L. 64849 MEXICO
52 (8) 328-4381
hma@campus.mty.itesm.mx

Abstract. This paper describes the working philosophy of the behavior-based architecture applied to the design of mobile robots. The main advantages with respect to the conventional architecture are presented together with the pioneering ideas of its creator. Finally I present the case of three different computer beings (softbots); they were programmed with the behavior-based architecture with different levels of "intelligence".

1. Introduction

The Artificial Intelligence is one of the most sounded disciplines in the last thirty years, sometimes most because its impressive name and the Hollywood culture than its real achievements. However in all this time many serious people have been studying and developing different approaches to solve a wide spectrum of problems [7]. All these approaches and theories since an operating standpoint can be seen as a sort of intelligence.

The AI community has suffered of the reduced knowledge about the human-thinking process, given that when a new artificial approach emerges, its success is limited into a specific domain.

So in the same way as the theoretical physician has not found a Great Unified Theory to explain the phenomena, both in the macro and the micro scale, the AI researcher has not been able to find a methodology capable to learn, to reason and to act in the real world without human intervention.

Before AI was applied to the robotics, the robot can be catalogued as a raw machine, there were not any significant difference between a wood drill and a welding robot arm, they carry out excellently its task only under the consideration in which they were designed or programmed, for instance the wood-drill still applies the same torque when we try to use it to drill concrete, and the robot arm couldn't apply the welding strip if it loses one of its sensors.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 136—147, 2000.

These types of robots only can operate in special working environment designed for them and praying for eternal well functioning of its sensors and actuators. We are living in a world built-up and inhabited by humans, this aspect makes the world especially difficult for the robot, because it has to deal with the inherently uncertainty of the environment and also with the unpredictability of the human beings.

In the case of the mobile robots, the variability of the working scenarios introduces more complexity to the problem, its task are not constrained at the same place, thus this kind of robots has to be specially skillful in order to carry out the task in spite of the environment.

The main aim of the AI in the robotics is to provide with the necessary robustness to the robot to operate in this ordinary world in an absolute autonomous way, working with the humans and in spite of the humans.

The architecture of a robot determines the form in which its resources are related. Concerning the mobile robots, traditionally they have been built-up in a hierarchical architecture, where is possible to identify a batch process in the processing of the information; first the robot has to get information of the environment through its sensors system, then a processing system extracts the necessary information from the data sensors, now the planning system can compute the necessary motion to achieve the goal, afterwards the execution system will produce the right motion commands to the actuators system, then the environment has been modified and the loop has been closed by the sensors system.

Each one of these systems requires of the correct operation of the preceding system. If a failure exists in whatever of these systems the loop is opened and the robot will go down.

It's easy to imagine what happens when the data sensor is corrupted either by noise or the poor accuracy of the sensor.

Hundreds of mobile robots with this architecture have been designed and probed its functionality only in engineered environment.

Brooks [1] broke with the hierarchical paradigm when he proposed a layered architecture [1], [2] and [3]. This new approach called Behavior-Based architecture was inspired in the insects. These organisms own a primitive degree of intelligence; even so they are able to survive at hostile environments. In the Behavior-Based architecture, the global performance of the robot emerges from the specific performance of every integrating system, unlike the hierarchical architecture, the sensor data is distributed parallel to each system; the way in which the system process the sensor data will define a single behavior, everyone of these behaviors incorporate some degree of competence, then when an external observer looks to the robot, he only perceives the final behavior.

This type of architecture provides of a very reactive performance in the robot achieving a high degree of robustness due the parallel distribution of the data, in this way if one of the systems is down the others behaviors still are working allowing to the robot achieve its task.

2. Behavior-Based Architecture

A mobile robot is an autonomous system able to operate in a working environment, trying to complete a task in spite of the uncertainty of the environment and the changing conditions.

Such a robot has to be able to recopilate information about the environment, extract the important characteristics of this data and process it according with the objective of the task, producing control commands, which operate the actuators of the robot, allowing the interaction with the environment. The way in which the robot achieves all this activities defines the architecture of the robot.

It seems that in thirty years of developing mobile robots, the researchers have maintained the same robot architecture, where the different modules are clearly separated and the signals flow is in a horizontal way as shown in Figure 1.

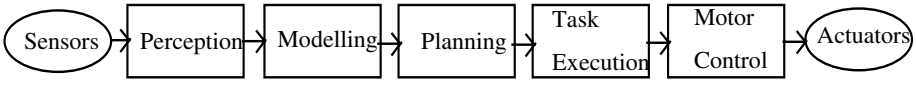


Fig. 1. Conventional robot architecture.

The main feature of this architecture type is that any module requires of the correct function of the precedent module, for instance, there will be a wrong planning if the modeling doesn't extract the right characteristics of the environment.

The main advantages of this architecture consists in the modeling-planning modules, because then the robot can create an internal model of the world which can be used to plan the motion of the mobile in an optimal way; but this implies to have accurate sensors and powerful data-processing aboard in order to get the best representation of the real world.

The data-processing module works on the basis of the symbol system hypothesis (the classical AI point of view), which imposes big limitations. In classical AI none of the modules themselves generate the behavior of the total system. There is an alternative dogma called nouvelle AI. It provides a different methodology for building intelligent systems, it decomposes the intelligence into individual behavior generating modules, whose coexistence and co-operation let more complex behaviors emerge[4].

The simplified architecture of such robot is shown in the Figure 2, where is possible to see the parallel flow of the data-sensors.

Such scheme defines a number of levels of competence for an autonomous mobile robot; a level of competence is an informal specification of a desired class of behavior [1] for a robot over all environments it will encounter. A higher level of competence implies a more specific desired class of behaviors.

The main idea of levels of competence is that its possible to build layers of a control system corresponding to each level of competence and simply add a new layer to an existing set to move to the next higher level of overall competence.

The system starts by building a complete robot control system which achieves level 0 competence, next another control layer is built, this is called the first level control system. This layer together with the level 0 achieves level 1 competence. The zero layer continues to run unaware of the layer above it which sometimes interferes with its data paths.

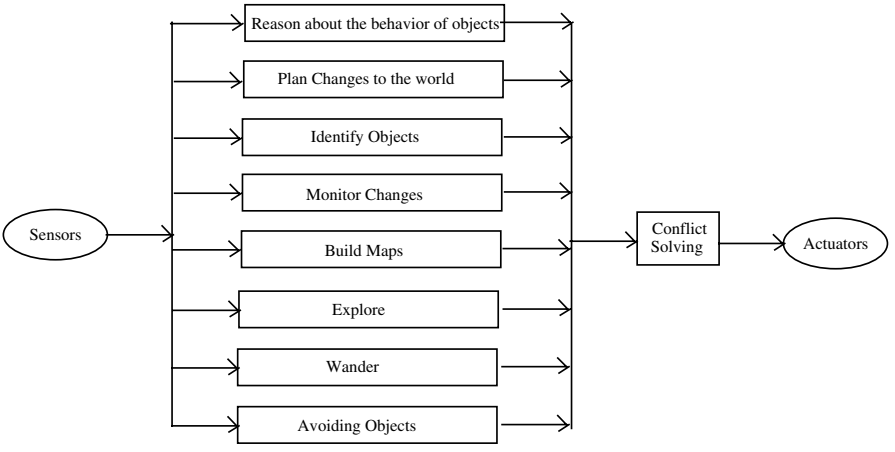


Fig. 2. Behavior-Based Architecture.

The same process is repeated in order to achieve higher levels of competence as the shown in Figure 3.

This type of architecture is called the subsumption architecture or the behavior-based architecture.

In such a scheme there is a working control system for the robot very early in the piece as soon as the first layer is built. Additional layers can be added later, and the initial working system need never be changed.

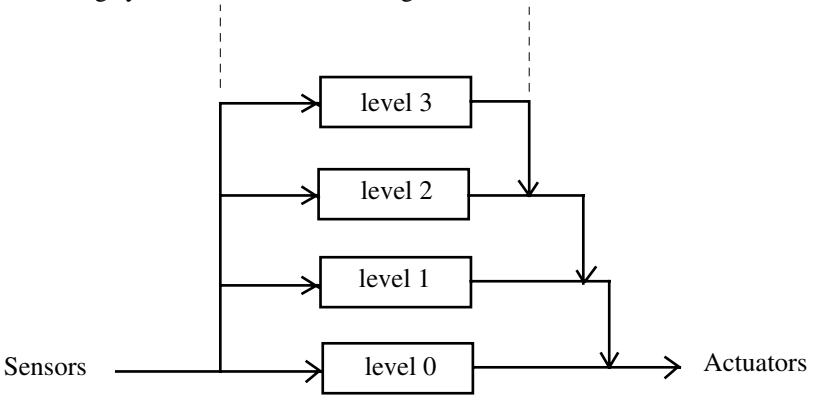


Fig. 3. Layered control in the subsumption architecture.

3. The Softbot

A softbot can be seen as a simulated robot working in a simulated environment, also can be considered as an intelligent agent [7]. The advantage to work with softbots are obvious, for instance, we don't have to be worry if the softbots hits against the wall at high velocity, is possible to test several times risky situations, we don't care to charge the batteries supply, etc.

The Behavior-Based architecture has been used to control a computer being which is able to navigate in a simulated environment where exist free-space and obstacles.

This software agent reminds the cockroach behavior, however is necessary to point out that the artificial agent exhibits a lower performance than a real cockroach, then the artificial agent can be seen as a proto-cockroach.

3.1. The Structure

The softbot resources are limited; they are integrated only by a sensor module, a processing unit, and a motion module; they together with its relationships are presented in Figure 4.

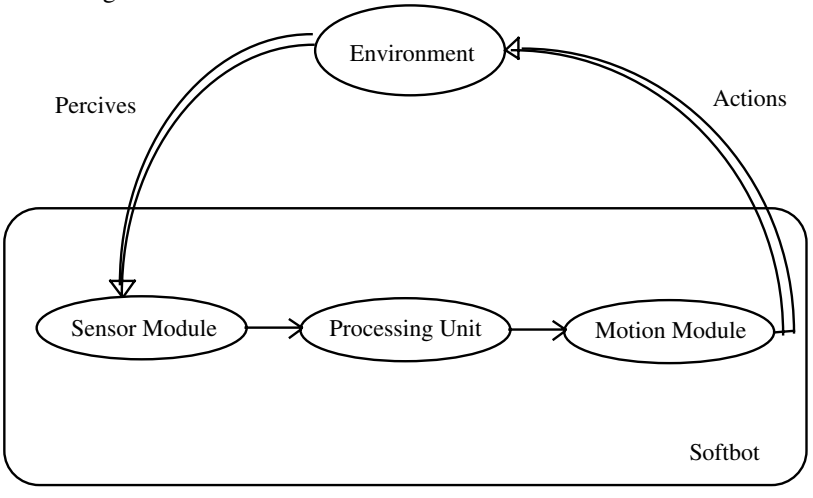


Fig. 4. Softbot structure.

The characteristics both the sensor module and the motion module depend of the softbot task, for instance if the robot only has to navigate in a working environment with solid objects embedded, the sensor module can be integrated by a set of sonar sensors and the motion module, a structure which allows the 2D displacement.

The look of the proto-cockroach is shown in Figure 5, which is a rigid body with a free displacement and five punctual sensors.

3.2. Sensor Module

The function of this module is to feedback the environment changes into the processing module, as shown in Figure 5, there are five punctual contact sensors aboard the softbot (S1... S5), all of them are binary sensor, that means when the end of the sensor is touching an edge wall, it turns its state to one, otherwise its state remains in zero.

The use of punctual sensors introduce a degree of risk in the robot navigation, because if there are acute or thin obstacles, they can pass between two sensors without avoid the collision; however, the use of punctual sensor is cheaper and emulates the whiskers of the cockroach.

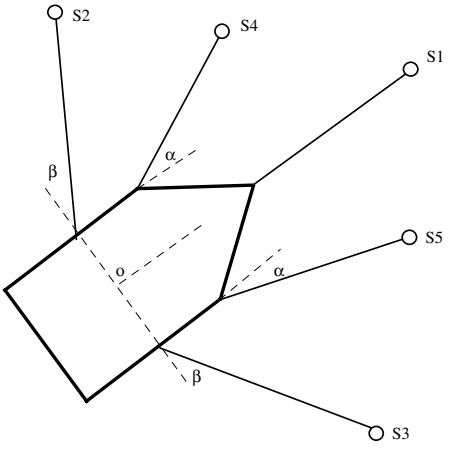


Fig. 5. The Softbot.

The arrangement of the sensor defines a safe threshold for the softbot; the body of the softbot can occupy all the space in this zone. The geometry of the safe zone depends on angles α and β. The S4, S1, and S5 sensors are used to avoid collisions against obstacles in the front of the softbot, whereas S2 and S3 are used to detect aside obstacles.

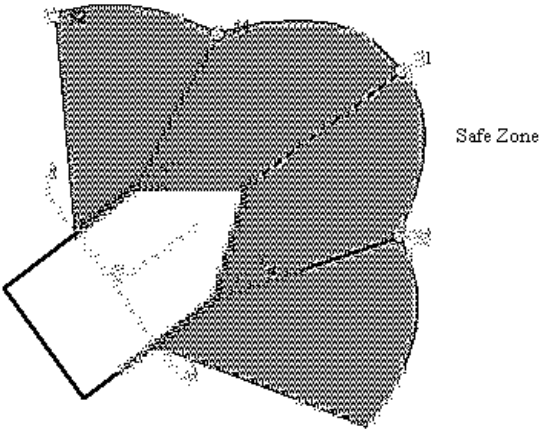


Fig. 6. Safe zone defined by the sensor arrangement.

3.3. Processing Unit

The processing unit is integrated by a set of defined behaviors, they are disposed in a layered configuration, the lower behaviors are responsible for the basic performance of the softbot and the higher corresponds to advanced and specific tasks. The structure of the processing unit for the present softbot is presented in the Figure 7.

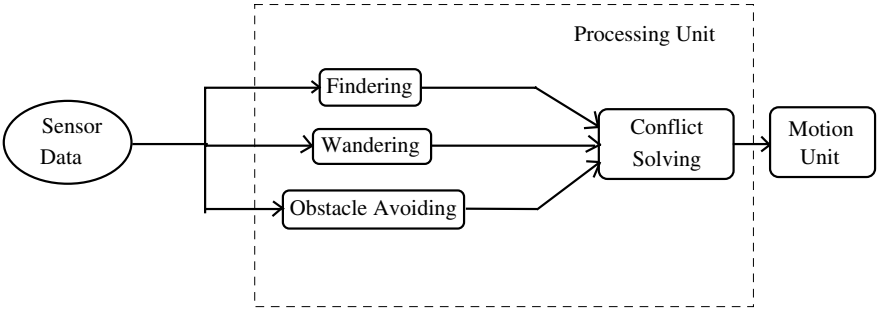


Fig. 7. Behavior-Based Architecture for the softbot.

Obstacle Avoiding Behavior

This behavior is responsible of avoid any collision with the obstacles wall, is implemented by a routine which uses as input the sensor data and produces an output motion command.

The behavior is integrated by a set of rules with the following syntax:

IF sensor-state (S1,...,S5) *THEN* motion = command

Where command is an order to control the motion of the softbot and will be described next. Sensor-state (S1,...,S5) is a Boolean function with the sensor state as input, this function defines the relative position of the softbot respect the obstacle.

In this case there are 2^5 possible sensed-positions for the robot.

The total set of rules programmed in the Obstacle Avoiding Behavior are written next:

IF (S2'·S4'· S5') *THEN* motion = Straight
IF (S1'·S2·S4' + S1'·S2·S3 + S1'·S3·S5 + S2'·S4'·S5 + S2·S3·S4') *THEN* motion = Right
IF (S2'·S4·S5' + S1·S2·S3' + S3'·S4·S5' + S1·S3'·S4 + S1·S4·S5') *THEN* motion = Left
IF (S1'·S3'·S4·S5 + S1·S3·S4·S5) *THEN* motion = Reverse

Where *Sn*' means no obstacle front the sensor and *Sn* obstacle front sensor.

In order to avoid the robot being trapped between an obstacle, a random selection of the direction is implemented with an occurrence probability of 0.25.

Wandering Behavior

It is not enough to have a softbot only controlled by the presence of obstacles, it has to be able to chose autonomously a direction and follow it, and when it finds an obstacle turn the direction avoiding the collision.

The wandering behavior programmed in the softbot, selects a direction in a random way once the softbot is safe, in order to avoid a chaotic motion, the direction is selected with some degree of probability trying to maintain a forward direction, then the probabilities distribution for the motion in the wandering behavior is presented in the Table 1.

Table 1. Command probaiblity occurrance.

Command	Probability
Right	1/7
Left	1/7
Reverse	1/7
Straight	4/7

When the wandering behavior is added to the softbot, there will be a bigger amount of explored space.

Target Searching Behavior

This top behavior has the objective to look for any target body, the target body is a screen object with special characteristics, and in this case the discriminative characteristic is the color, a color different than the wall obstacle color.

To achieve this behavior is necessary to add a sensor able to distinguish the color, three-color sensors have been placed at the same position of the sensor S4, S1, S5, but the new sensor has a bigger scope.

Then the new behavior is integrated by a set of rules, which guide the softbot direction towards the target, as shown below:

IF SS1 *THEN* motion= Left
IF SS2 *THEN* motion= Straight
IF SS3 *THEN* motion= Right

This reduced instruction set is designed to work when the target extension is smaller than the angle between sensors.

Conflict solving unit

Since every behavior produces an output command, sometimes these orders could be opposed, then it is necessary to include a module which can choose the best option to the softbot.

Obviously the "best option" is a relative expression because it will depend of the global behavior (something equivalent to the personality of the robot) that we wish to incorporate to the softbot.

The behaviors of the present softbot are programmed in the following way: first the softbot will try to be out of danger of collisions against obstacles, once the softbot is safe it can start a wandering activity at the same time if the softbot detects a target it will change its direction towards the target.

The task of the conflict-solving unit is precisely to maintain this relationship between all the behaviors.

Motion Unit

This unit receives as input a motion command and will traduce this order into real displacement of the softbot body. There are two types of displacement: the translational and the rotational motion. A combination of this type will produce the total displacement of the softbot. Table 2 shows all commands allowed.

Table 2. Allowed commands.

Command	Action
Right	Turn to the right
Left	Turn to the right
Reverse	Linear reverse displacement
Straight	Linear forward displacement

4. Results

The results are presented in a gradual way: behaviors will be added step by step in order to compare the change in the "personality" of the softbot.

Figure 8 shows the performance of the softbot controlled only by the obstacle-avoiding behavior, clearly the walls of the obstacles guide the softbot.

Figure 9 corresponds to the softbot performance exhibited when the obstacle avoiding behavior, together with a wandering behavior are programmed. The motion of the softbot is freer and it can visit more space in the working environment.

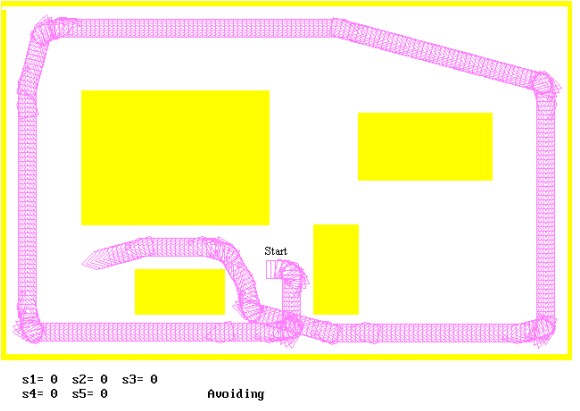


Fig. 8. Obstacle avoidance behavior.

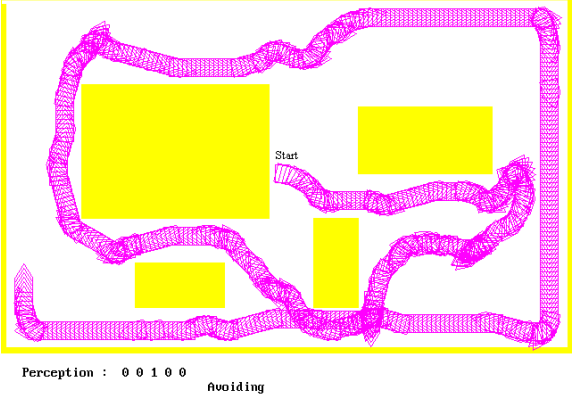


Fig. 9. Avoidance and wandering behavior.

A third behavior has been included in the softbot, such behavior allows to the softbot identify a target and change its direction towards the target; Figure 10 shows the followed path of the softbot achieving the three static targets without collision.

Figure 11 shows an emerging behavior when the target is mobile, the softbot try to follow it taking care all the time of the obstacles.

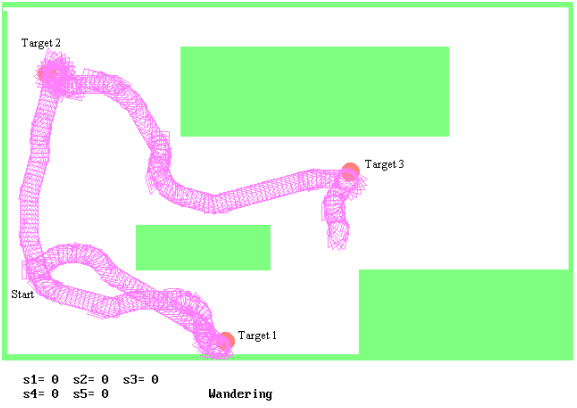


Fig. 10. Three working behavior.

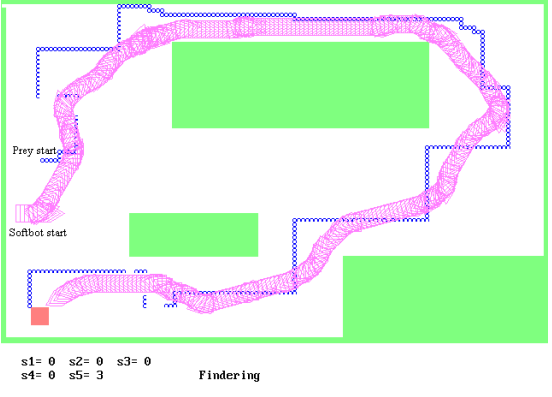


Fig. 11. The hunter emerging behavior.

5. Conclusions

The behavior-based architecture allows the implementation of very reactive robots; the reactivity is a necessary characteristic of a robot, which is working in a dynamic world interacting with static obstacles, dynamic bodies and people.

The parallel processing of the data-sensor gives to the robot an well-appreciated robustness due the independence of its modules to work, unlike the conventional architectures, when one of the robot modules is damaged, the robot still can work.

The behavior-based architecture provides of a very basic degree of intelligence to the robot similar to the insects, in spite of the reduced intelligence, these small organisms are able to achieve complex task as the building of its houses, the food supply, reproduction, and so on.

It is a fact that a robot with a low degree of intelligence will have a constrained application field, however is possible to combine the strengthens of the behavior-based architecture with the modern reasoning and visual processing approaches [6], achieving more complete intelligent machines.

We think that before humans build complex robots with anthropomorphic behaviors, first it is necessary to study simpler organism in order to catch the secrets of the intelligent reactivity and adaptability to the world.

Bibliography

[1] Brooks Rodney A. *A robust layered control system for a mobile robot.* AI. Memo 864 Massachusetts Institute of Technology. September 1985

[2] Brooks Rodney A. *The behavior language; users guide.* AI. Memo 1227. Massachusetts Institute of Technology. April 1990

[3] Brooks Rodney A. *Achieving artificial intelligence through building robots.* AI. Memo 899. Massachusetts Institute of Technology. May 1986.

[4] Brooks Rodney A. *Elephants don't play chess.* Robotics And Autonomous Systems 6. 1990.

[5] Brooks Rodney A. *New approaches to robotics.* Massachusetts Institute of Technology.

[6] Mataric M.J. *Integration of representation into goal-driven behavior based robots.* Robotics and Automation IEEE Trans. June 1992

[7] Norvig and Russell. *Artificial Intelligence. A Modern Approach.*

Cooperative Simulated Annealing for Path Planning in Multi-robot Systems

Gildardo Sánchez-Ante, Fernando Ramos, and Juan Frausto

Department of Computer Science
ITESM Campus Morelos
Av. Paseo de la Reforma 182-A
Cuernavaca, Morelos, 62589
MEXICO

Abstract. This paper presents some preliminary results on the application of a Cooperative Simulated Annealing algorithm (COSA) for solving the problem of path planning in Multi-Robot systems. The main idea is to generate paths for each robot in the system without taking care of the other robots, and then coordinate the paths for all the robots. Paths are generated using a variation of the probabilistic roadmaps (PRM), and the coordination of the paths is achieved with a Cooperative Simulated Annealing (COSA). To evaluate the system, several experiments for environments constituted by two planar robots with up to five dof and various convex obstacles were run. The results obtained aim to continue working on this line.

1 Introduction

Multi-Robot Motion Planning (MRMP) is usually seen as an extension of the basic problem on motion planning. Simply stated, MRMP consists in generating safe trajectories for a set of robots that share their working spaces. Trajectory planning for multiple-robot systems fall into two main approaches: centralized and decoupled methods.

Centralized approaches treat the separate robots as one composite system, and typically perform the planning in a composite configuration space, formed by combining the configuration spaces of the individual robots. Some important results in this approach are, for instance, those obtained by Shwartz and Sharir who developed an exact cell decomposition algorithm for multiple discs in the plane moving among polygonal obstacles. The complexity of the algorithm is $O(n^3)$ for 2 discs, and $O(n^{13})$ for 3 discs [6], [7]. Some other interesting results were obtained by Barraquand and Latombe applied the randomized path planner (RPP) to multi-robot planning problems [1]. Perhaps the main advantage of the RPP is its ability to escape from local minima. Potential fields have been also explored as an alternative. Specifically, Barraquand, Langlois and Latombe studied its application to problems with narrow corridors [2].

Decoupled approaches first generate paths for the separate robots more or less independently, and then consider the interactions between the robots (with

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 148–157, 2000.

respect to the generated paths) and coordinate the movements of the robots in such a way that collisions are avoided. Erdmann and Lozano-Perez proposed the scheme of *prioritized planning* [3]. In this case, each robot receives a priority. Each robot is picked in an order given by its priority, and a path for it is found. Such path avoids collisions with the static obstacles as well as with the previously picked robots, which are considered as moving obstacles. O'Donnell and Lozano-Perez introduced a decoupled planning approach called *path coordination*, which is based on a scheduling technique originally developed by Yannakakis et al. for concurrent database access by several users [5]. First, paths for the two robots are planned independently, then a two dimensional coordination diagram is constructed and used for finding those configurations in which the robots collide. The diagram is used for detecting the conflictive configurations, and then the paths can be coordinated by modifying the velocities of the robots.

The algorithm proposed in this work is based on the decoupled approach, in the sense that paths are generated independently for each robot, but a second step would remind the coordination approach, given that the obtained paths are coordinated to avoid collisions between robots. In other words, the algorithm proceeds in two steps:

a) *Path generation*: Robots are picked one at a time, and a path is found using an efficient probabilistic roadmap planner, proposed by Hsu and Latombe [4]. It must be pointed out that we do not consider the previously generated paths as a constraint for the other robots. The paths are generated considering just the static obstacles.

b) *Multiple path coordination*: Each path is taken and a set of local modifications for the original path is generated. Once we have that set, we apply an optimization technique named Cooperative Simulated Annealing (COSA), in order to find an adequate combination of paths (i.e., a set that enables each robot to accomplish its movements without colliding with the rest of the robots and the static obstacles)[8].

The paper is structured as follows: Section two describes an algorithm for Cooperative Simulated Annealing, section three establish the MRMP problem in terms of an optimization approach, and describes our algorithm. In section four the experiments and results are presented, and finally, section five is devoted to the conclusions as well as to describe some future work.

2 Cooperative Simulated Annealing

Simulated annealing is a non-deterministic optimization technique based on a physical process, named annealing, which is used to produce low-energy molecular arrangements in metals. Simulated annealing has been used for finding low-cost solutions to combinatorial problems. The algorithm behaves as a greedy algorithm that sometimes chooses a worst solution with a probability given by the Boltzmann distribution. This characteristic enables the algorithm to escape from local minima. The algorithm for combinatorial optimization problems can be summarized as follows:

1. Start with a first candidate solution S_c. It can be randomly generated, or can be extracted from context knowledge. The solution S_c has an associated cost W_d, and it is the best solution so far (and is also the only one we have).

2. Apply a perturbation to S_c, in such a way that we generate a modification in the neighborhood of S_c. It can be, for instance, a permutation to S_c. Let name S_t the newly generated solution, and let W_t denote its cost.

3. If $W_t < W_d$ take S_t as the best solution (throwing away the one we had before) and go to step 6.

4. Else if $W_t \geq W_d$ generate a random number x such that $(0 < x < 1)$.

5. If $x < \exp[(W_t - W_d)/T]$, accept the proposed solution S_t (even when it has a higher cost). T is a parameter to regulate the acceptance ratio (the greater T, the lower probability of accepting a worst solution). In the physical process T is the temperature. This step is used to escape from local minima.

6. Iterate steps 2-5 until a certain number of iterations is reached or another stopping criteria is satisfied.

The main advantage of the method is its ability to escape from local minima, but it generally requires a huge number of transitions for obtaining satisfactory results. A more detailed description on Simulated Annealing can be found in [10], [11].

In order to decrease the number of transitions, Wendt proposed a combination of Simulated Annealing (SA) and Genetic Algorithms (GA). The main idea is to exchange information within a population apart from crossover operators, performing smaller jumps on the fitness landscape, thus needing less repair information and finding a more efficient way towards the global optimum. The result was named COSA, a hybrid local search algorithm [8], [9].

COSA inherits the idea of population and information exchange from Genetic Algorithms but replaces the usual crossover by the so-called cooperative transitions. The acceptance probability for the transition is controlled by the Metropolis function, depending on the virtual variable called temperature and the difference in the solutions' goal function values (as usually done in SA).

COSA thus implements a concurrent but synchronous run of multiple SA processes, loosely coupled by the cooperative transitions. These cooperative transitions replace the usual uniform probability distribution PG_{ij} (the probability of generating neighbor j from solution i) by a skewed distribution, which relates the direction of the (potential) search step to the current position of other individuals in the population.

A more formal description of the algorithm is:

1. Initialize population size $|POP|$, where POP is the set of individuals (solutions).

2. Generate initial population

$POP_0 = \left(I_1 \in S, I_2 \in S, ..., I_{|POP|} \in S\right) \in S^{|POP|}$, where each I_i is an individual (solution), and S is the set of all valid solutions..

3. Initialize neighborhood relationship $N \subseteq S \times S$

4. Initialize the number of transitions $K = n \times |POP|_{k-1}, n \in N$

5. Set the initial temperature $T_1 \in R_+$ and the cooling rate $\alpha \in]0; 1[\subset R_+$

6. **for** $(k = 1$ to $K/|POP|)$

{ 7. **for** ($i = 1$ to $|POP|$)
{
8. $I_j \in POP_{k-1}$
9. $I_{new} = cotrans(I_i, I_j)$
10. **if** $random(0,1) \leq \begin{cases} 1 & \text{when } f(I_{new}) < f(I_i) \\ e^{-\frac{f(I_{new})-f(I_i)}{T_k}} & \text{when } f(I_{new}) \geq f(I_i) \end{cases}$
11. **then** $I_i = I_{new}$
}
12. $POP_0 = \left(I_1, I_2, ..., I_{|POP|}\right)$
13. T_{k+1}=update$(POP_{k-1}, POP_k, T_k, \alpha)$
}
14. **end**

where

function update$(POP_k \in S^{|POP|}, POP_{k-1} \in S^{|POP|}, T \in R_+, \alpha \in]0;1[) \rightarrow T \in R_+$

$\Delta E = E(POP_{k-1})$, where E represents energy (fitness).

$T = \begin{cases} T & \text{when } \Delta E < 0 \\ \alpha T & \text{when } \Delta E \geq 0 \end{cases}$

return T

Function 'update' compares the mean energy (fitness) of the current population POP_k with the mean energy of the ancestor population POP_{k-1}: If it increased, temperature T gets multiplied by the annealing factor (e.g. 0.999), otherwise T is kept constant. The rationale behind this procedure is the following: Whenever T gets decreased it takes some time for the population to reach the thermodynamical equilibrium determined by the new temperature. If the population is big enough, a rising mean energy is a good criterion for having reached this equilibrium, and now starting to randomly fluctuate around this equilibrium (although there may of course be stochastic rises in mean energy before having reached the equilibrium). The step 10 is the one responsible of the most important characteristic of the Simulated Annealing: the escape from local minima. If the fitness of the new individual $f(I_{new})$ is lower (when minimizing) than the fitness of the current individual $f(I_i)$, then the probability of accepting the new solution is 1, otherwise, the probability is given by the Boltzmann function.

3 Multi-robot Motion Planning with COSA

Let suppose that we have two manipulator robots (R_1 and R_2), arranged in such a way that they share part of their corresponding working spaces, which means that collisions can occur. The problem consists in finding safe paths for both robots. There could also be static obstacles in the environment.

The solution proposed in this work can be stated as follows:

a) Generate a path for each robot (consider just the static obstacles).

b) Once we have both paths, check if they can be executed without collision between the robots. If it is the case, the problem is solved. Otherwise, we need

to coordinate the paths. The coordination is done by using the Cooperative Simulated Annealing.

It is necessary to give more details on both steps:

3.1 Path-Generation Step

For the path-generation step we chose an implementation of Probabilistic Roadmaps (PRM) proposed by Hsu and Latombe [4]. Probabilistic path planning consists in randomly build a dense roadmap that can be later used for point-to-point fast path planning. It has been applied successfully to a variety of robot tasks as manipulation by a robot arm having a high number of degrees of freedom, path planning for non-holonomic car-like robot, and coordination of the motion of multiple (car-like) robots. The problem is defined by a certain preprocessing of the configuration space ($\mathcal{C}$-space), after which many difficult path planning problems can be solved in time of the order of a fraction of a second. The preprocessing itself usually does not take very long: of the order of a few hundreds of seconds. During the preprocessing stage, a set of collision-free configuration (nodes) in the $\mathcal{C}$-space are generated and interconnected into a network using very simple and fast planning techniques applied to pairs of neighboring nodes. The network produced has a large number of nodes (order of thousands). It may contain one or more components, depending on the robot's free space and the time spent on preprocessing.

After preprocessing, planning a path between any two configurations is solved by connecting both configurations to some two nodes A and B in the network, and searching the network for a sequence of edges connecting A and B. The resulting path can be improved using any standard smoothing algorithm. The planner fails if it cannot connect any of the two input configurations to the network, or if A and B lie in two different connected components of the network.

The approach described before is very useful in the case of completely static environments, in which the computational effort devoted to the construction of the roadmap is exceeded by the saving in time when answering a query (i.e., a path). However, there are cases in which the environment is evolving, and in which it would not be practical to construct a PRM. Hsu, Latombe et al. proposed an alternative for those cases [4]. The main idea is to iteratively grown a tree rooted at the initial configuration towards the final configuration. The algorithm starts with the initial configuration q_i, and randomly generate a set of configurations in the vicinity of q_i. It then tries to connect those configurations to q_i, if the connection is possible it adds an edge from the current configuration to q_i in the tree. Then, a configuration from the tree is picked and the same process is applied (i.e. generate configurations in the neighborhood, try to connect them to the tree, etc.). Figure 1 illustrate the process.

3.2 Path-Coordination Step

The first step is used to generate one valid path for each robot. Certainly there is no warranty that the set of paths generated is valid (remember that during

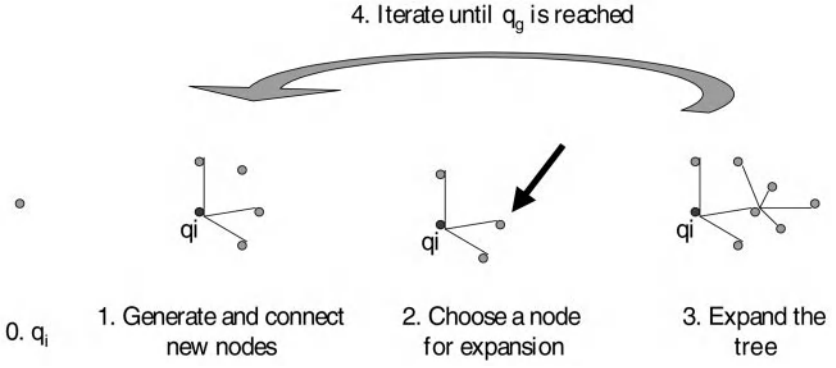


Fig. 1. The process of generation of a path for each robot.

the first step only the static obstacles were considered for the generation of the paths). The following process tries to coordinate the paths by applying an optimization technique named Cooperative Simulated Annealing (COSA). The basic idea is to generate a set of modifications to the paths (which corresponds to a *population* in Genetic Algorithms) and try to find the best path. Of course that the path must not have any collision, and it can also optimize other parameters. In the next step, we apply COSA, as described in the last section. Figure 2 shows the complete process.

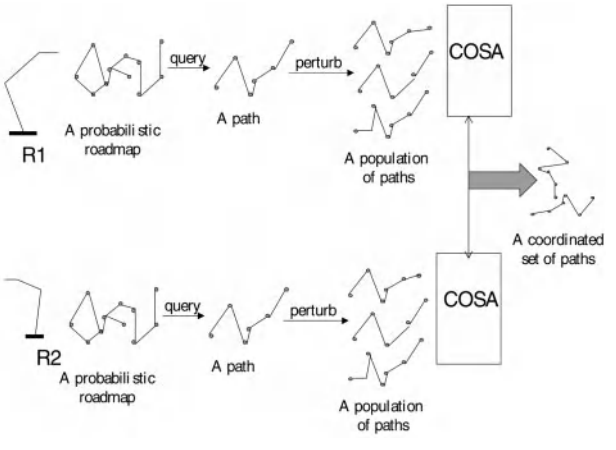


Fig. 2. A schematization of the application of a Cooperative Simulated Annealing to the Multiple-Robot Motion Planning Problem.

It is very important to point out a couple of points on the application of COSA to the MRMP problem:

- Due to the nature of the implementation of COSA that we took (the one of Wendt), there is interchange of information between paths in the same population. The rationale behind using a Cooperative Algorithm, instead of a simple simulated annealing is that the interchange of information accelerates the convergence of the algorithm.
- However, it is not enough for solving the problem. It is required that the paths do not provoke any collision between robots. For getting that, it is necessary to coordinate the movements of the robots. One possible way in which it can be done is by defining an objective function (fitness function) in terms of the cooperation that the robots are doing for:
  - Avoiding collision with the other robots.
  - Cooperating to facilitate the advance of the other robots. This feature acquires more importance in cases when one of the robots reach its goal but when doing so block the movement of the other robot. We would like that the first robot check if there is other position that it could take that do not block the second robot.

Operator Definitions As usual when using genetic algorithms, it is necessary to define the genetic operators to be used. In this case, the crossover and the mutation operators are defined.

Crossover To illustrate the way in which each operator is applied, let consider two paths:

A={117 82 103 31 117}, {35 33 114 88 9}, {10 40 119 28 52}, {33 26 16 6 63}

B={117 82 103 31 117}, {111 23 62 8 30}, {12 49 91 102 88}, {118 92 19 5 23}

It is possible to generate a new path, by combining parts of the above two paths, for instance:

C={117 82 103 31 117}, {111 23 62 8 30}, {12 49 91 102 88}, {33 26 16 6 63}

It is important to remark that given that all the configurations in the path are free, the newly generated path is constructed also from free configurations. However, that is not enough to ensure that the path itself is free. It is necessary to call at a local planner to make sure that all the steps of the path are in the free-space.

Mutation For the case of the mutation operator, the implementation chosen in this work is to randomly pick a configuration in the path, and change it for another which is also free. For example:

If the original path is P_1 = {117 82 103 31 117}, {35 33 114 88 9}, {10 40 119 28 52}, {33 26 16 6 63}, then a mutation could be P_1' = {117 82 103 31 117}, {35 33 114 88 9}, {7 93 60 27 6}, {33 26 16 6 63}

Of course that there could be many other ways of defining both operators, and the way in which they are implemented may affect the performance of the algorithm. Some of them could be explored in further works.

4 Experimental Work

All the experiments were accomplished on simulation. The robots considered are planar with 5 degrees of freedom. There are several static obstacles. All of them are represented as convex polygons. Figure 3 shows a typical configuration of the environment in the simulator.

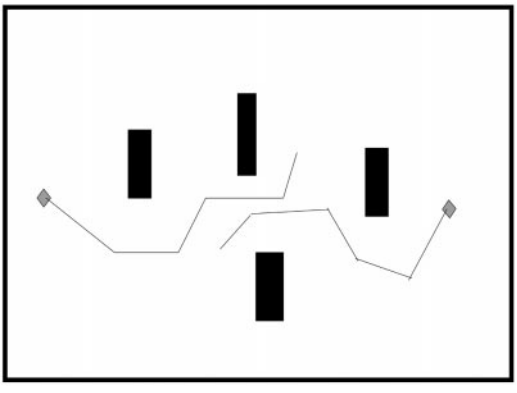


Fig. 3. A simulated environment for testing the algorithm. Black boxes represent static obstacles.

The interval for each joint angle is discretized to 128 values. However, kinematic constraints are taken into account. The algorithm respects the limits of each joint.

The first step of the algorithm (the generation of one path for each robot) was run on a Sun Ultra SPARC-IIi with 128 Mb RAM and the COSA was run on a IBM RS/6000 model 250 with 34 Mb RAM. The codes were implemented in C. In each case the initial path was composed by 1000 configurations.

In the table shown above, the second column presents the time required for each robot to find one valid path. It should be pointed out that the paths are not optimal. Each experiment correspond to a different set of initial-final configurations. The number and size of the obstacles was kept constant.

Table 1. Results from the experiments run

Experiment	Time Step 1 (s)	Time Step 2 (s)	Total Time (s)
1	152.4 + 125.3	874.0	1151.7
2	220.3 + 122.0	665.2	1007.5
3	188.7 + 189.9	890.3	1268.9
4	141.2 + 211.7	733.4	1086.3
5	120.9 + 105.1	580.6	806.6
6	121.1 + 202.3	956.2	1279.6
7	203.7 + 103.2	1098.5	1405.4

5 Conclusions and Future Work

What was presented in the last section corresponds to a very limited set of experiments. Our goal was to explore the feasibility of the methodology. In that respect, it is possible to conclude from the results that the approach seems promising.

It is possible to observe that the second step of the methodology took more time than the first one. It is due mainly to the way in which paths are modified. There are two points in which the paths suffer from modifications. The first one is when generating the population for COSA. In that case, given a path, a cluster of five points is changed by other randomly generated. The second point for path modification occurs in inside COSA, in which case the modifications are given by the genetic operators described before. Obviously with this approach, the probability of having a good guess varies inversely proportional with the number of robots in the environment. The idea was taken from [15], but there could be a better way of doing so. In the current version the dynamics of the robot is not taken into account, but if we would like to consider it, the problem of coordinating the paths will increase in complexity.

The normalization of the values of each angle facilitated the work of generating new configurations. It was equivalent to generate a set of five integer values in the interval from 0 to 128.

It is our intention to continue working on this problem, specifically on points like:

- In the case of continuing exploring on the current implementation of the planner, it could also be interesting to try some alternative definitions of the genetic operators. Even with the same definitions, there could be some parameters to adjust. For example: in this case, the crossover takes always clusters of 10 configurations to be exchanged. No experimentation was done modifying that value and checking its influence on the performance of the algorithm.
- To substitute COSA by an algorithm developed with the MRMP problem in mind. There are some related works done by people at the ITESM Morelos that we would like to explore [14], [12], [13].
- To accomplish a set of experiments with different environments, varying the number of robots and their degrees of freedom, the number of obstacles and their positions, the amount of free space, etc.
- To establish a comparison between this approach an some of the others described in the introduction. It would be very interesting trying to establish the influence of the variables on the performance of the algorithm.

References

1. Jerome Barraquand and Jean-Claude Latombe: Robot Motion Planning: A Distributed Representation Approach Int. Journal Robotics Research 10(6), 1991, 628–649

2. Jerome Barraquand, and Bruno Langlois, and Jean-Claude Latombe Numerical Potential Field Techniques for Robot Path Planning, 1992 IEEE Trans. on Systems, Man and Cybernetics, Vol. 22, No.2, 224–241
3. M. Erdmann, and R. Lozano-Pérez On Multiple Moving Objects, IEEE Int. Conf. Robot. Automat., 1986, 1419–1424
4. David Hsu, and Jean-Claude Latombe, and R. Motwani Path Planning in Expansive Configuration Spaces Proc. IEEE Int. Conf. On Robotics and Automation, 1997
5. P. A. O'Donnell and T. Lozano-Perez Deadlock-Free and Collision-Free Coordination of Two Robot Manipulators IEEE Int. Conf. Robot. Automat., 1989, 484–489
6. J. T. Schwartz, and M. Sharir On the Piano Movers' Problem I: The Case of a Two-Dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers Communications Pure and Applied Mathematics, 1983, (36), 345–398
7. J. T. Schwartz, and M. Sharir On the Piano Movers' Problem II: General Techniques for Computing Topological Properties of Real Algebraic Manifolds Advances in Applied Mathematics, 1983, (4), 298–351
8. Oliver Wendt COSA: Cooperative Simulated Annealing, Tech. Rep. Number 94-09, 1994 Universitaet Frankfurt
9. Oliver Wendt Naturanaloge Verfahren Zur Approximativen Lösung Kombinatorischer Optimierungsprobleme Universität Frankfurt, 1995
10. V. Cerny Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm Journal of Optimization Theory and Applications, 45:1, 1985,41–51
11. S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi Optimization by Simulated Annealing, 1983, (2220),671–680.
12. F. Ramos, and G. Gomez, and G. Sanchez-Ante, Cooperative randomized methods for planning motions in mono and multiple robotic systems, 1999, Proc. Primer Congreso de Robótica, Asociación Mexicana de Robótica.
13. Héctor A. Montes Planificación de trayectorias para múltiples robots manipuladores usando algoritmos genéticos Tesis de Maestria, ITESM Campus Morelos, 1999.
14. V. de la Cueva, and F. Ramos Cooperative Genetic Algorithms: A New Approach to Solve the Path Planning Problem for Cooperative Robotic Manipulators Sharing the Same Workspace Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 1 pp. 267-272, Victoria, B.C. Canada, October 1998.
15. G. Sanchez-Ante, and F. Ramos, and J. Frausto Locally Optimal Path Planning by using Probabilistic Roadmaps and Simulated Annealing Proc. IASTED Robotics and Applications International Conference, Santa Barbara, CA, 1999.

Learning Probabilistic Grid-Based Maps for Indoor Mobile Robots Using Ultrasonic and Laser Range Sensors

Leonardo Romero, Eduardo Morales, and Enrique Sucar

ITESM, Campus Morelos, Temixco, Morelos, 62589, Mexico
lromero@zeus.ccu.umich.mx
{emorales,esucar}@campus.mor.itesm.mx

Abstract. A new method for learning probabilistic grid-based maps of the environment of a mobile robot is described. New contributions on the three major components of map learning, namely, sensor data fusion, exploration and position tracking, are proposed. In particular, new models of sensors and a way of sensor data fusion that takes advantage of multiple viewpoints are presented. A new approach to control the exploration of the environment taking advantages of local strategies but without losing the completeness of a global search is given. Furthermore, a robot position tracking algorithm, based on polar and rectangular correlations between laser range data and the map is also introduced. Experimental results for the proposed approach using a mobile robot simulator with odometer, ultrasonic and laser range sensors (implemented with laser pointers and a camera), moving in an indoor environment, are described.

1 Introduction

The ability of a mobile robot to learn a model of the environment, using its sensors, increases its flexibility and usefulness. This ability can be used for dynamic or new environments without requiring a re–programming effort. This is particularly useful for mass production of service's robots in office environments or museums. The problem of building a model of an environment by a mobile robot, while it is moving is difficult and far from being solved [17]. The following factors impose practical limitations [17]:

- *Sensors.* Sensors often are not capable of directly measuring the quantity of interest.
- *Perceptual limitations.* The perceptual range of most sensors is limited to a small range around the robot.
- *Sensor noise.* Sensor measurements are typically corrupted by noise.
- *Drift/Slippage.* Robot motion is inaccurate and odometric errors accumulate over time.

A model of the environment used for navigating, is commonly called a *map*. This work considers full metric maps with square regions of the same size. Regi-

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 158–169, 2000.

ons have occupancy probabilities associated to them. This type of map is commonly called *Probabilistic Grid-based Maps* or PGM, [2],[8],[11], [16],[18]. This type of map is appealing because it can be easily used to fuse sensor data.

This work also considers a mobile robot with a ring of ultrasonic range sensors, or *sonars* for short, a common arrangement used in other previous research [17]. The robot has as well a laser sensor, which measures proximity of nearby objects using a combination of several laser pointers and a camera. Laser pointers are parallel to the floor in a radial arrangement. Distances to objects can be estimated considering the height of the *laser points* within images.

The proposed approach to build a PGM can be described by three major components:

Sensor data fusion. Multiple sensor data are mapped and integrated over time to occupancy values of grid cells. An extended approach to the sonar data fusion described in [3] is given. A new laser range model is also presented. Finally, a way to fuse data from different sensor types is described.

Exploration. The robot explores its environments trying to reach the nearest unexplored grid cell minimizing the travel cost. A novel approach to merge local strategies, like *wall following*, within a global search frame, based on a dynamic programming algorithm, is given.

Position estimation. The position of the robot is continuously tracked, minimizing odometric errors. This paper proposed a new approach to estimate the position of the robot based on correlations between laser range data and the map.

The remainder of this paper is organized as follows. Section 2 describes the approach to sensor data fusion. The description of the exploration strategy is given in Section 3. Section 4 describes the proposed approach to the robot position tracking. Experimental results using a mobile robot simulator are presented in Section 5. Finally, some conclusions and future research directions are given in section 6.

2 Sensor Data Fusion

The environment is modeled as a set of cells arranged in a regular two–dimensional grid. The occupancy of a cell at location (x,y) is measured by the state variable $O(x, y)$ which can have one of two values: *occ* (occupied) and *free*. For each cell, the probability that $O(x, y) = occ$, denoted by $P(O(x, y) = occ)$ is estimated. In this work it is considered that the robot has three types of sensors:

Ultrasonic range sensors. In this case, let $P(O_s(x, y) = occ)$ be the occupancy probability of the cell (x,y) detected only by sonars.

Laser range sensors. In the same way, let $P(O_l(x, y) = occ)$ be the occupancy probability detected only by this type of sensors.

Maneuverability. The mere fact that a robot moves to a location (x,y) makes it unlikely that this location is occupied. Let $P(O_m(x, y) = occ)$ be the occupancy probability detected by this type of sensor.

A cell will be considered occupied if it is detected occupied by at least one sensor. In this way, the probability that a given cell is occupied can be estimated using a logical operation OR among the states of occupancy detected by each type of sensor:

$$P(O(x,y) = occ) = P([O_s(x,y) = occ]\, OR\, [O_l(x,y) = occ]\, OR\, [O_m(x,y) = occ]) \tag{1}$$

To expand the right hand side of (1), it is assumed that the events $O_s(x,y) = occ$, $O_l(x,y) = occ$ and $O_m(x,y) = occ$ are statistically independent. With this assumption, and after a little of algebra, equation (1) becomes:

$$P(O(x,y) = occ) = 1 - \prod_{i=s,l,m} (1 - P(O_i(x,y) = occ)) \tag{2}$$

This expression can be used to compute the probability that a cell is occupied once we have determined the probability that a cell is occupied by each type of sensor. The prior probabilities that $P(O(x,y) = occ)$ are initially set to 0.5 to indicate ignorance. This implies that the prior probabilities for the variables associated to each type of sensor i $(i = s, l, m)$ for every cell are given by:

$$P_{prior}(O_i(x,y) = occ) = 1 - (0.5)^{1/3} \tag{3}$$

A description of how to compute the probability that cells are occupied using each type of sensor is described in the following sections.

2.1 Ultrasonic Range Sensors

There are two main difficulties using sonars: 1) they have a wide beam, like a 30 degrees cone and 2) a specular reflection occurs whenever an ultrasonic pulse encounters a smooth extended surface. In ordinary office environments which contain smooth walls and glass doors specular reflection is common [3]. Elfes [2] and Moravec [11] describe an approach in which range measurements from multiple viewpoints are combined into a PGM. Each cell in the grid is assigned a *single* value indicating the probability that the cell is occupied. Unfortunately, the occupancy grid approach does not work well in specular environments [3]. Howard and Kitchen [3] propose an improvement of grid-based approaches by introducing the concept of *response grid*. The basic idea is that a cell may generate a response (e.g. appears to be occupied) when viewed from one direction, but will not generate a response when viewed from another.

Following the approach described in [3], when an ultrasonic pulse entering a cell with some direction can be reflected back to the detector, the cell is said to have a *response* in that direction. The occupancy value of the cell is determined by assuming that any cell that generates a response in one or more directions must contain at least one surface and therefore be occupied. The response of a cell (x,y) in some direction ϕ is measured by the variable R (res means response): $R(x,y,\phi) = res$. The full interval $[0, 360^o]$ is divided into n

intervals. Let $R_i(x,y) = R(x,y,\phi_i)$ be the response for the direction interval ϕ_i. The probability that the cell is detected occupied using sonars is given by:

$$P(O_s(x,y) = occ) = P([R_1(x,y) = res]\ OR\ \cdots\ OR\ [R_n(x,y) = res]) \quad (4)$$

To expand the right hand side of (4), it is assumed that the events $R_i(x,y) = res$ are mutually independent. With this assumption, (4) becomes:

$$P(O_s(x,y) = occ) = 1 - \prod_{i=1}^{n}(1 - P(R_i(x,y) = res) \quad (5)$$

Applying Bayes rule to determine the probability $P(R_i(x,y) = res)$, given a range measurement r:

$$P(R_i(x,y) = res|r) = \frac{P(r|R_i(x,y) = res)P(R_i(x,y) = res)}{P(r)} \quad (6)$$

The prior probability $P(R_i(x,y) = res)$ can be computed from equation (3):

$$P_{prior}(R_i(x,y) = res) = 1 - (1 - P_{prior}(O_s(x,y) = res))^{\frac{1}{n}} \quad (7)$$

The other important term in equation (6) is the *sonar sensor model* $P(r|R_i(x,y) = res)$. The sensor model indicates the probability of obtaining the measurement r, given that proposition $R_i(x,y) = res$ is true. Let s be the distance between the cell (x,y) and the sensor. The sensor model proposed in [3] is:

$$P(r|R_i(x,y) = res) = \begin{cases} 0.05 & \text{if } s < r \\ 0.5 & \text{if } s > r \\ \alpha/r & \text{if } s = r \end{cases} \quad (8)$$

where α is a normalization constant (summing over the probabilities assigned to each cell at range r should yield a total probability of 1).

The sensor model of equation (8) proposed in [3] has the following problems. In some cases, for an r and s given, $P(r|R_i(x,y) = res)$ could be smaller than 0.5 when $s = r$. The limit value should be 0.5 when r is huge. Another difficulty arises if we need to update $P(R_i(x,y) = res)$ given two or more measurements.

The following approach is proposed. To compute $P(R_i(x,y) = res)$ given m sensor readings, denoted $r^{(1)}$,$r^{(2)}$, ... ,$r^{(m)}$, one has to assume conditional independence between $r^{(i)}$ and $r^{(j)}$ $(i \neq j)$ given that $(R_i(x,y) = res)$ is true. In other words:

$$P(r^{(i)}|R_i(x,y) = res, r^{(1)}, \ldots, r^{(i-1)}, r^{(i+1)}, \ldots, r^{(m)}) = P(r^{(i)}|R_i(x,y) = res) \quad (9)$$

With this assumption, a probabilistic update rule for $P(R_i(x,y) = res)$ can be deduced [16], [17]:

$$P(R_i(x,y) = res|r^{(1)}, \ldots, r^{(m)}) = 1 - [1 + \frac{P(R_i(x,y) = res)}{1 - P(R_i(x,y) = res)} Prod_1]^{-1} \quad (10)$$

where

$$Prod_1 = \prod_{j=1}^{m} [\frac{P(R_i(x,y) = res|r^{(j)})}{1 - P(R_i(x,y) = res|r^{(j)})}][\frac{1 - P(R_i(x,y) = res)}{P(R_i(x,y) = res)}]$$

This update rule is frequently used for the accumulation of sensor evidence [11],[12]. This equation can be used to update the probabilities in an incremental way. Using this update rule, a sensor model $P(R_i(x,y) = res|r)$ should satisfy the following constraints:

- When $s > r$ then $P(R_i(x,y) = res|r) = P_{prior}(R_i(x,y) = res)$. In this way, the probability of a cell whose distance is greater than the sonar reading r is not changed.
- When $s = r$ then the probability $P(R_i(x,y) = res|r)$ should be in the interval $[P_{prior}(R_i(x,y) = res), Ps_{max}]$ according to the value of r. It must have values close to 1 (Ps_{max}) for small values of r and values close to $P_{prior}(R_i(x,y) = res)$ for large values of r. In other words, short sonar readings tend to significantly increase the probability of occupancy, while long sonar readings tend to slightly increase it.
- When $s < r$ then $P(R_i(x,y) = res|r)$ should be in $[Ps_{min}, P_{prior}(R_i(x,y) = res)]$ according to the value of r. It must have values close to 0 (Ps_{min}) for small values of r and values close to $P_{prior}(R_i(x,y) = res)$ for large values of r. In other words, short sonar readings tend to significantly decrease the probability of occupancy, while long sonar readings tend to slightly decrease it.

The following sensor model satisfies these constraints:

$$P(R_i(x,y) = res|r) = \begin{cases} 1 - (1 - P_{prior}(R_i(x,y) = res))(1 - Ks_{max})^{\frac{1}{Nc}} & \text{if } s < r \\ P_{prior}(R_i(x,y) = res) & \text{if } s > r \\ P_{prior}(R_i(x,y) = res) Ks_{min}{}^{\frac{1}{Nc}} & \text{if } s = r \end{cases} \tag{11}$$

where Ks_{max} is a constant close to 1, Ks_{min} is another constant close to 0, and Nc is the number of grid cells ($Nc >= 1$) at range s covered by the sonar.

Notice that this model ignores statistical errors associated with the range value itself. However, in practical applications, these errors are insignificant compared with errors arising from other sources, such as localization errors.

2.2 Laser Range Sensor

A camera together with laser pointers parallel to the floor are aligned to implement laser range sensor. This arrangement can rotate to cover the whole area around the robot. For instance, in the experiments, for each position of the robot, 16 images with 8 laser points per image are obtained, giving a total of 128 range data around the robot.

Let us now consider the uncertainty in the estimated distances due to image resolution. Let h be the distance between the x–axis of the image and the laser

point within the image, H be the height of the laser point (a constant), f be the camera focal length and Z be the distance from the camera center to the laser point (the object) (see Fig. 1 (a)). The relative error of Z due to image resolution (Δh) is given by [15]:

$$|\frac{\Delta Z}{Z}| = \frac{1}{h}|\Delta h| \tag{12}$$

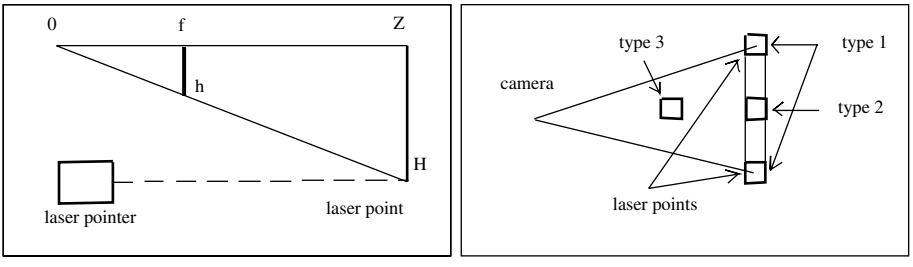


Fig. 1. Laser model. (a) Laser range sensor implemented with laser pointers and a camera. (b) Types of cells within the laser model.

Considering similar triangles, the absolute error $E_l = |\Delta Z|$, for a given distance Z, can be expressed as:

$$E_l(Z) = K_l Z^2 \tag{13}$$

where $K_l = \frac{\Delta h}{fH}$ is a constant that depends on the specific camera (f and Δh) and the vertical distance between the camera and the laser pointers (H). In order to update the probability $P(O_l(x, y) = occ)$ given a set of laser range data, the proposed model considers the laser data in consecutive pairs[1]. Three types of cells (see Fig. 1 (b)), for two consecutive readings are considered:

1. Cells associated to each reading.
2. Cells between cells of type (1), considering a linear interpolation.
3. Cells inside the polygon formed by the camera and the two readings.

For each type of cell, the probability $P(O_l(x, y) = occ)$, given two consecutive readings $Z^{(1)}, Z^{(2)}$ is updated in the following way. Let $Z^{(i)}$ be the reading associated to one cell of type (1). Then, the probability that this type of cell is occupied given the reading is given by:

$$P(O_l(x, y) = occ|Z^{(i)}) = 1-(1-P_{prior}(O_l(x, y) = occ))(1-Kl_{max})^{E_l(Z^{(i)})} \tag{14}$$

[1] That is, adjacent laser points on an angular sequence.

where Kl_{max} is a constant close to 1. This expression takes into account the absolute error associated to the reading, and assigns high values to short readings and low values to large readings. The update rule for cells of type 1, chooses the minimum value between the old and the new probability values. In this way, short laser readings overwrite probabilities due to longer readings. To update cells of type 2, if the minimum of $P(O_l(x,y) = occ)|Z^{(1)})$ and $P(O_l(x,y) = occ)|Z^{(2)})$ is greater than the old value, then choose the minimum value between $P(O_l(x,y) = occ)|Z^{(1)} + Kl_a)$ and $P(O_l(x,y) = occ)|Z^{(2)} + Kl_a)$. This means that short readings, as before (considering the minimum of both readings), overwrite probabilities due to longer readings. Here Kl_a depends on the belief that these cells are occupied. This value could be a function on how uniform is the part of the image that contains the two laser points. To update cells of type 3, if the minimum of $P(O_l(x,y) = occ)|Z^{(1)})$ and $P(O_l(x,y) = occ)|Z^{(2)})$ is greater than the old value, then multiply the old value by a factor Kl_i in the interval $(0,1)$. Kl_i depends on the belief that these cells are free.

This is a much simpler approach that equation (10) that can be used in this case since laser range data are less noisy than sonars.

2.3 Maneuverability

In this case, we use a simple update rule. To update $P(O_m(x,y) = occ)$ given the position of the robot, all the cells under the robot are decreased by a factor Km in the interval $(0,1)$.

3 Exploration

To autonomously acquire maps, the robot has to explore. The general idea is to let the robot always move on a minimum–cost path to the nearest unexplored grid cell [17]. The minimum–cost path is computed using modified versions of *value iteration*, a popular dynamic programming algorithm. In [17] the cost for traversing a grid cell is determined by its occupancy value, while in [10] the cost is determined by the distance between cells (see chapter 8 in [6]). This paper proposes an approach to merge local search strategies within a modified version of value iteration described in [10]. When the robot starts to build a map, all the cells have a probability of occupancy $P(O(x,y) = occ) = 0.5$. A cell is considered *unexplored* when its probability is close to 0.5, defined by two constants $[Pe_{min}, Pe_{max}]$, and *explored* in other case. In contrast, in [17] a cell is considered explored when it has been updated al least once. That approach, however, does not work well when there are specular surfaces in the environment, since multiple measurements are normally required, in this case, to find the obstacles around the robot.

Cells are defined as *free* or *occupied*. A cell is considered occupied when its $P(O(x,y) = occ)$ reaches a threshold value Po_{max} and continues to be occupied while its $P(O(x,y) = occ)$ does not fall below a threshold value Po_{min} (where

$Po_{min} < Po_{max}$). It is considered *free* in other case. This mechanism prevents changes in the state of occupancy of a cell by small probability changes.

In this work, a cylindrical (circular base) robot was used, so the configuration space (c–space) can be computed growing the occupied cells by a ratio of the robot. In fact, the c-space is extended to form a *travel space*. The idea behind the travel space is to define a way to control the exploration following a kind of *wall following strategy*. Wall following is a local method that has been used to navigate robots in indoor environments, but unfortunately it can be easily trapped in loops. The travel space together with a dynamic programming technique has the advantages of both worlds, local and global strategies. We consider that a travel space splits the free cells of the c-space in three categories:

1. Close to an occupied cell in the c-space. Let D_w be the maximum distance between a cell and its closest occupied cell. These cells are called *warning cells* because their purpose is to warn the robot about its closeness to an obstacle. Let C_w be the set of these cells.
2. Close to a warning cell. Let D_t be the maximum distance between a cell and its closest warning cell. These cells are called *travel cells* because their purpose is to suggest to the robot a path to follow. Let C_t be the set of these cells.
3. *Far cells*. Any free cell in the c–space that is not a warning or a travel cell, will be a far cell. Let C_f be the set of these cells.

A travel space can be computed incrementally after each change of state of a cell (occupied or free), while the map is being build. An example of a travel space is shown in the Figure 2.

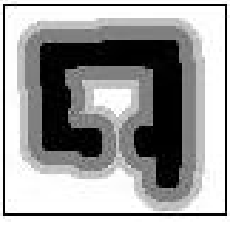


Fig. 2. A travel space. From darker to lighter: occupied cell (black), warning cells (dark gray), travel cells (light gray), and far cells (white)

A policy to move to the unexplored cells following minimum–cost paths is computed using a modified version of value iteration and a travel space. The algorithm uses two variables, V and M, associated to each cell. $V(x, y)$ denotes the travel cost from cell (x,y) to the nearest unexplored cell. $M(x, y)$ represents the optimal movement to choose, given that the robot is in that cell. The valid movements are 8, one per cell in its vicinity. If $M(x, y) = (dx, dy)$, where dx is the change in x and dy is the change in y, the set of valid movements is

$M_v = \{(1,0),(1,1),(1,0),(-1,1),(-1,0),(-1,-1),(0,-1),\ (1,-1)\}$. The idea is to associate costs to cells depending on its type. If warning cells and far cells have costs higher than travel cells, then a wall following strategy of exploration is taken into account, instead of a greedy strategy.

Using these variables, the algorithm has two steps:

1. Initialization. Unexplored cells (x, y) that are free in c–space are initialized with $V(x, y) = 0$, all the others free cells with ∞. All the free cells in c-space are initialized with $M(x, y) = (0, 0)$.

2. Update. Let (x_r, y_r) be the position before the last movement of the robot. For all the explored free cells $(x, y) \neq (x_r, y_r)$ in c–space do:

$$V(x,y) \leftarrow min_{(dx,dy)\in M_v}\{V(x+dx, y+dy) + Cost((x,y),(dx,dy)\} \quad (15)$$

$$M(x,y) \leftarrow arg\_min_{(dx,dy)\in M_v}\{V(x+dx, y+dy) + Cost((x,y),(dx,dy)\} \quad (16)$$

where $Cost((x, y), (dx, dy))$ measures the cost of moving from the cell (x, y) to the cell $(x+dx, y+dy)$. This function punishes changes in direction and is given by:

$$Cost((x,y),(dx,dy)) =$$

$$\begin{cases} C(x+dx,y+dy) + Dist((x,y),(x+dx,y+dy)) & \text{if } (dx,dy) = M(x+dx,y+dy) \\ C(x+dx,y+dy) + Dist((x,y),(x+dx,y+dy)) + Kp_c & \text{if } (dx,dy) \neq M(x+dx,y+dy) \end{cases} \quad (17)$$

Where $Dist(p_1, p_2)$ is the distance between cells p_1 and p_2 (1 or $\sqrt{2}$), and Kp_c represents the cost of changing directions. $C(x, y)$ represents the cost associated to cell (x, y), in the travel space, based on its type. Let Kp_w, Kp_t and Kp_f be the costs associated to warn, travel and far cells.

The update rule is iterated and when the values $(V(x, y)\ ,M(x, y))$ converge, the robot execute the movement indicated by M. The exploration does the following general steps:

1. Process the readings taken by all the sensor and update the probability of occupancy of the cells.
2. Choose the next movement using value iteration.
3. Do the movement.
4. Get readings from sensors and correct odometric error (see next section).
5. Go to the first step.

Exploration ends when $V = \infty$ for the cell where the robot is placed, which means that there is no way to reach an unexplored cell. Each time the value iteration algorithm is called, only the V values around the robot are initialized, in a similar way to the *bounding box* described in [17].

The robot can occasionally move very close to a transparent surface, like a glass door, since laser range sensors fails to detect the surface and sonars can not detect it until it is too close, due to specular reflections. In this case the robot can even be inside an occupied cell. To deal with this case, a list of movements of the robot is kept and used to backtrack the robot to the last free cell position.

4 Position Tracking

Position tracking is the problem of estimating the robot position while it is moving, using its sensors. As noted in [16], position tracking is particularly difficult to solve if map learning is interleaved with localization. A recent survey in [1] dedicated to this topic illustrates the importance of localization and illustrates the large number of existing approaches. The approach proposed in this paper is a version of model matching using metric maps. If differs from previous approaches in that take it considers only laser range data and the map in construction. The position tracking algorithm computes the actual location of the robot in two steps:

1. Rotation. From the last position of the robot a *laser visibility view* is computed from the map, using only the probabilities associated to the laser range sensors ($P(O_l(x, y))$. For reference, this view will be called a *map view*. Then, the map view is converted to a *polar map view* taking the robot position as the center, and using linear interpolation to get a smooth curve. In a similar way, a *polar sensor view* is obtained from the current laser range data (*sensor view*). The rotation between the last position and the new position can be estimated using a correlation between the polar map view and the polar sensor view.

2. The position of the robot can be estimated using a correlation between the map and the sensor view, once the sensor view has the same orientation that the map view. In this correlation only the sensor readings above some threshold value Kd are considered.

5 Experimental Results

This section presents the results obtained using a mobile robot simulator. The laser range sensors were implemented using equation (13). See [14] for details about the sonar model implemented. Figure 3 (a) shows a map built using the three sensors when the position tracking mechanism was not used. The robot has an uniform random error on displacements of $\pm 10\%$ and a uniform random orientation error of about ± 7 degrees, per movement. It shows the occupancy probability of map cells. The map is about $10\text{x}10m^2$ and the grid cells are $10\text{x}10cm^2$ Note that errors in the odometer are too high and the map built is not useful at all. Figures 3 (b) and (c) show the maps built using the position tracking mechanism, with $Kd = 300cm$ and $Kd = 200cm$ respectively (Kd is the threshold distance value used in the correlation to calculate the displacement of the robot). In these two cases: $Dw = 50cm$, $D_t = 20cm$, $Kp_w = 8$, $Kp_t = 0$ and $Kp_c = 3$. These values implement the wall following strategy to explore the environment, as can be observed in the maps. The travel space associated to the map of Fig. 3 (c) is shown in Fig. 3 (d). The position tracking approach assumes that the robot movements are small enough, so that the maps before and after processing sensor data, look alike. The actual position tracking algorithm considers possible displacements of one grid cell. When the robot returns to a partially known place, after a long trip, the error position could be bigger than one cell. In fact, the map of Figure 3 (b) shows this event.

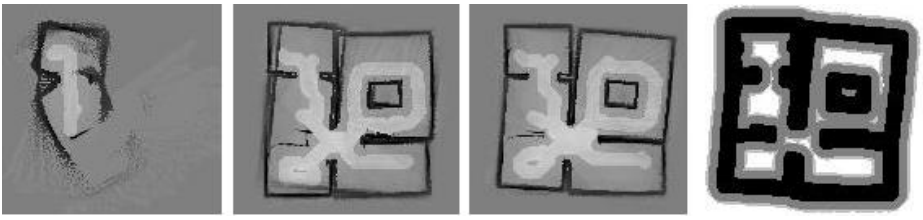


Fig. 3. Maps built considering that the robot has odometric errors. White areas represent cells with occupancy probabilities near to 0. From left to right. (*a*) Without using the position tracking algorithm. (*b*) Using the position tracking algorithm with $Kd = 300cm$ (threshold distance value used in the correlation to calculate the displacement of the robot). (*c*) Using the position tracking algorithm with $Kd = 200cm$. (*d*) The travel space associated to (c)

We also did some experiments to observe the effect of changes due to the cost of making orientation changes in the robot movements (Kp_c). These results suggest that higher Kp_c values tend to decrease the number of movements that change the orientation of the robot. The effect of Kp_c is analog to the effect of *inertial mass*: it tends to keep the orientation (velocity in the other case) of the robot.

6 Conclusions

A new method of map learning for indoor mobile robots using ultrasonic, and laser range sensors was presented. This paper extends the approach described in [3] to fuse ultrasonic range data, gives a new probabilistic model of laser range sensors, and presents a way of sensor data fusion of different types. The experimental results show that this approach is a working method to fuse sensor data. Additionaly, a new approach to explore the environment that combines local control with a global exploration strategy has been described. As experimental results confirm, a sort of wall following technique is taken into account during exploration. This combination of local and global strategies takes the advantages of both worlds: robustness of local ones, and completeness of global ones. A heuristic to minimize the number of orientation changes, trying to minimize the accumulated odometric error is also introduced. This heuristic implements a kind of *inertial mass* of the robot. Finally, a new position tracking approach using correlations between the laser range data and the map was also introduced. The experimental results show that the proposed approach works even when very high odometric errors are considered.

We plan to explore some dynamic extensions to the travel space approach considering the specular degree of cells, captured by the sonar model. Testing the approach using real mobile robots is also considered.

References

1. Borenstein, J., Everett, B., and Feng, L.: Navigating Mobile Robots: Systems and Techniques. A.K. Peter, Ltd., Wellesley, MA. (1996).
2. Elfes, A.: Using Occupancy Grids for Mobile Robot Perception and Navigation. IEEE Computer, 22(6) (1989) 46–57.
3. Howard, H., y Kitchen, L.: Generating Sonar Maps in Highly Specular Environments. Proceedings of the Fourth International Conference on Control, Automation, Robotics and Vision, December (1996).
4. Kuipers, B. and Byun, Y.T.: A robust qualitative method for spatial learning in unknown environments. In Proceedings of Eighth National Conference on Artificial Intelligence AAAI-88, Menlo Park, Cambridge, AAAI Press/The MIT Press (1988).
5. Kuipers, B. and Byun, Y.T.: A robust exploration and mapping strategy based on a semantic hierarchy of spatial representations. Journal of Robotics and Autonomous Systems 8 (1991) 47–63.
6. Lee D.: The Map–Building and Exploration of a Simple Sonar–Equipped Robot. Cambridge University Press (1996).
7. Leonard, J.J., Durrant-White, H.F. and Cox, I.J.: Dynamic map building for an autonomous mobile robot. International Journal of Robotics Research, 11(4) (1992) 89–96.
8. Lim, J.H. and Cho, D.W. : Physically Based Sensor Modeling for a Sonar Map in a Specular Environment. Proc. IEEE/RSJ International Conference on Robots and Systems. (1992) 62–67.
9. Matarïc, M.J.; A distributed model for mobile robot environment–learning and navigation. Master's thesis, MIT, Cambridge , MA. (1990).
10. McKerrow, P.J.: Introduction to Robotics. Electronic Systems Engineering, Addison-Wesley (1991).
11. Moravec, H.P.: Sensor Fusion in Certainty Grids on Mobile Robots. AI Magazine 9(2) (1988) 61–74.
12. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers, San Mateo, CA., (1988).
13. Rencken, W.D.: Concurrent localization and map building for mobile robots using ultrasonic sensors. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Yokohama, Japan, (1993) 2129–2197.
14. Romero, L. and Morales, E.: Uso de una red neuronal para la fusion de lecturas de sonares en robots moviles. Segundo Encuentro Nacional de Computacion (ENC99), Pachuca, Hidalgo, Mexico, (1998).
15. Shigang, L., Ichiro, M., Hiroshi, I., and Saburo T.: Finding of 3D Structure by an Active–vision–based Mobile Robot. Proceedings of the IEEE International Conference on Robotics and Automation (1992).
16. Thrun S., Bucken, A., Wolfram B. et al.: Map Learning and High-Speed Navigation in RHINO. En Kortenkamp, D., Bonasso, R.P., y Murphy, R.: Artificial Intelligence and Mobile Robots. AAAI Press/The MIT Press (1998).
17. Thrun S.: Learning Maps for Indoor Mobile Robot Navigation. Artificial Intelligence, 99(1), (1998) 21–71.
18. Yamauchi, B., Schultz, A., Adams, W., and Graves, K.: Exploration and Spatial Learning Research at NCARAI. Conference on Automated Learning and Discovery, June 11-13, (1998) Carnegie Mellon University, Pittsburg, PA.

Evolving Insect Locomotion Using Cooperative Genetic Programming

Edgar E. Vallejo[1] and Fernando Ramos[2]

[1] Computer Science Department,
ITESM Campus Estado de México Apdo. Postal 50,
Módulo de Servicio Postal Campus Estado de México del ITESM,
52926 Atizapán, Edo. México, México,
evallejo@campus.cem.itesm.mx
[2] Graduate School of Informatics, ITESM Campus Morelos
Ave. Paseo de la Reforma 182 Col. Lomas de Cuernavaca, 62589
Cuernavaca, Morelos, México,
framos@campus.mor.itesm.mx

Abstract. In this work, genetic programming systems are viewed as cooperative processes: individuals in the population *cooperate* to evolve a global solution. There are two basic forms of cooperation for distributed problem solving: task-sharing and result-sharing. We introduce two models that enable cooperation in genetic programming systems. The first model is based on task-sharing and the second one is based on result-sharing. We examine the effects of both forms of cooperation on the performance of genetic programming systems solving the insect locomotion problem. This article demostrates that cooperative genetic programming can be used to evolve several commonly observed gaits in insects.

1 Introduction

The problem of insect locomotion has been approached with several methods. Brooks [3] has used a subsumption architecture in building a robot that walks. Beer [2] has used a recurrent neural network to control locomotion in a simulated insect. Spencer [13] has demostrated that genetic programming (GP) can be used to evolve locomotion with a minimum of assumptions about how walking should occur.

Some insect inspired locomotion models suggest that locomotion may be viewed as an emergent property of the cooperation between the mechanisms responsible for the control of individual legs [7]. We used this assumption for building a model of cooperative genetic programming (CGP). The model evolves programs that give a simulated insect the ability of walking.

Traditionally, evolutionary computation systems have been viewed as competitive processes: individuals in the population *compete* with each other to evolve a global solution. The interaction between individuals is governed by the principle of natural selection. More recently, evolutionary computation systems have

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 170–181, 2000.

been viewed as cooperative processes: individuals in the population *cooperate* with each other to evolve a global solution [4], [10].

Research in distributed artificial intelligence has focused for many years on the cooperative solution of problems by a collection of agents. Smith and Davis [11] have classified cooperation models for distributed problem solving in two main categories: task-sharing and result-sharing. In task-sharing, agents share the computational workload, executing one or more of the subtasks the overall problem has been split into. In result-sharing, agents share their own partial results, each reflecting the independent perspective that an individual has of the overall problem.

We introduce two models to enable cooperation in genetic programming systems. This first model is based on task-sharing and the second one is based on result-sharing. In the task-sharing model, each program consists of several subprograms for the execution of subtasks of the overall problem. In the result-sharing model, individuals in the population accept or refuse to share partial results using a decision procedure. We exploit the structure of genetic programming with automatic definition of functions [9]: function-defining branches are viewed as partial results. Individuals in the population decide to cooperate with a potential partner on the basis of expected payoffs using a decision matrix [8].

We demostrate that both forms of cooperation can be used to evolve several commonly observed gaits in insects as emergent behaviors. We conclude with a discussion of possible improvements to this work.

2 Insect Locomotion

There is a growing interest in using insect locomotion systems to control walking robots. Insect locomotion is notably robust and flexible. Insects can walk over a variety of terrains. In addition, they can also adapt their gait to the loss of up to two legs without serious degradation of performance.

Insect locomotion systems must solve the problems of *support* and *progression*. The problem of support arises because the gravitational attraction of the earth must be overcome. The problem of progression arises because an insect must generate propulsive forces that overcome both the inertia of the body and the friction of the substrate. To provide support and progression, the movement of the legs must be coordinated by the locomotion system.

Ferrell [7] has defined several terms associated with insect locomotion:

- *Protraction:* The leg moves towards the front of the body.
- *Retraction:* The leg moves towards the rear of the body.
- *Power stroke:* The leg is on the ground where it supports and then propels the body. In forward walking, the leg retracts during this phase.
- *Return stroke:* The leg lifts and then swings to the starting position of the the next power stroke. In forward walking, the leg protracts during this phase.
- *Anterior extreme position (AEP):* In forward walking, this is the target position of the return stroke.
- *Posterior extreme position (PEP):* In forward walking, this is the target position of the power stroke.

Wilson [15] developed a model for characterizing most of the commonly observed gaits of insects, including those resulting from amputation. Some of these gaits are shown in figure 1.

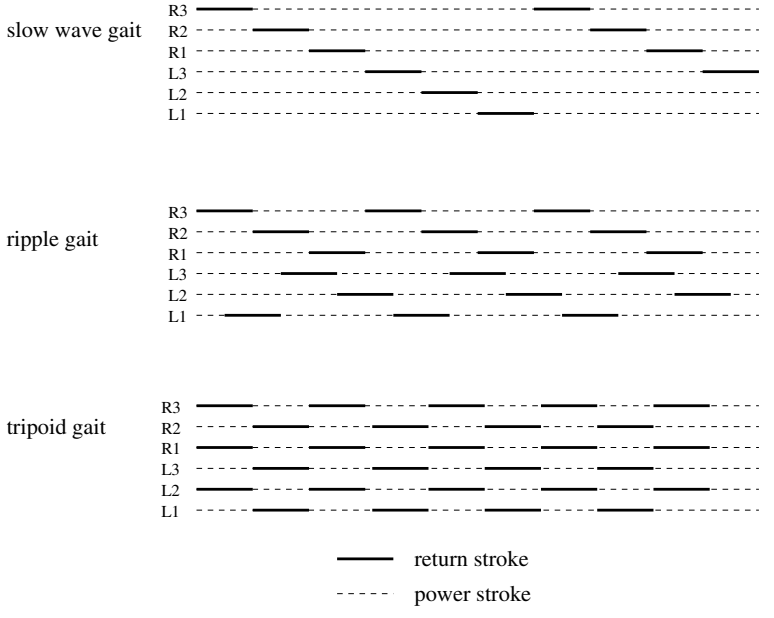


Fig. 1. Commonly observed gaits [15]

Cruse [5] developed a model that employs stimulus and reflexes to generate leg motion and gait coordination. There are three basic mechanisms responsible of the coordination between legs. They are shown in figure 2. These mechanisms affect the threshold for beginning the return stroke by adjusting the posterior extreme position of the receiving leg.

In this model, locomotion is an emergent property of the cooperation between the mechanisms responsible of the control of individual legs. We used this assumption for building a model of cooperative genetic programming to evolve locomotion as emergent behavior.

3 Cooperative Genetic Programming

Underlying most social systems there are many instances of cooperation. Individuals of the same species live together in groups for their mutual benefit. Similarly individuals of different species make associations when cooperation is potentially to their mutual advantage.

Dawkins [6] has shown that genes, when viewed as agents, can cooperate to mutually improve their chances of reproducing. Similarly, Axelrod [1] has

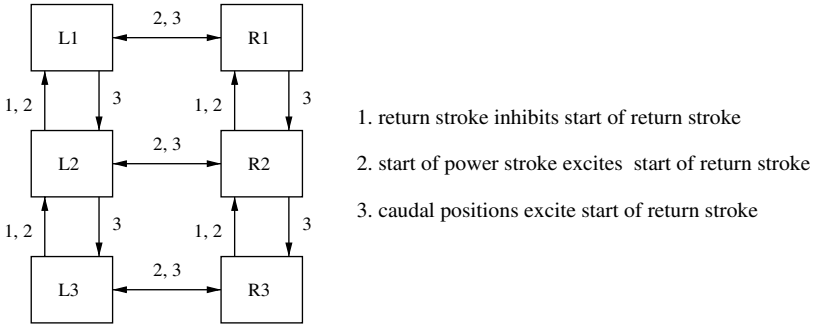


Fig. 2. Cruse's model for leg coordination [5]

identified attributes of agents that succesfully cooperate when confronted with the prisoner's dilemma game.

In evolutionary computation systems, the transmission of information between individuals is through reproduction: parents transmit their genes to offsprings. It has been shown that the performance of genetic programming systems can be enhanced through the transmission of non-genetic information. Spector and Luke [12] used a shared indexed memory mechanism for the transmission of non-genetic information between contemporary and non-contemporary individuals.

Cobb [4] has previously analysed the behavior of genetic algorithms using a cooperative learning paradigm. This work demostrated that the performance of genetic algorithms can be improved through result-sharing using a blackboard architecture. Similarly, Luke and Spector [10] have shown how teamwork and coordination can be evolved in genetic programming systems.

We present two models for cooperation in genetic programming systems based on task-sharing and result-sharing [14], respectively. In the task-sharing model, each program consists of several subprograms for the execution of subtasks of the overall problem. In the result-sharing model, individuals in the population accept or refuse to share partial results using a decision procedure.

3.1 Task-Sharing in Genetic Programming

We made two modifications to the basic genetic programming to enable task-sharing. The first modification is to the program structure and the second one is the design of restricted genetic operators appropriate to the program structure.

Task-sharing in genetic programming requires the existence of a decomposition of the overall problem into subproblems in such a way that the integration of the individual solutions for subproblems results in a solution for the overall problem.

Program Structure. Suppose that there exists a decomposition of an overall problem into n subproblems. Then, each program in the population consists of n subprograms, each subprogram representing a solution to each subproblem.

The syntactical structure of the program is augmented to include n *S-expressions*. For each *S-expression*, we define a function set, a terminal set and program characteristics appropriate for the solution to each subproblem.

Genetic Operators. The genetic operators have been designed to preserve the syntactical structure of the individuals in the population. The crossover operator must be performed between subprograms that solve the same subproblem to preserve the syntactic validity of all offspring. Similarly, mutation operator should consider the function set and terminal set defined for each subprogram to be mutated.

3.2 Result-Sharing in Genetic Programming

We made two modifications to the genetic programming with automatic definition of functions to enable result-sharing. The first modification is to the program structure and the second one is the addition of a cooperation phase between generations.

Program Structure. We exploit the features of functional programming for the modification of the program structure. In functional programming, functions are treated as *first class objects*: functions can be passed as arguments, returned as the value of other functions and can be bounded to any arbitrary variable. Functions that accept functions as arguments or return functions as values are called *higher order functions*.

In our implementation, the overall program is defined as a *lambda expression* with one argument, the variable `RS`. This argument, called the *result-sharing variable*, is bounded to a procedure during execution. This is the interface that enables individuals to interact one another by sharing procedures: the overall program of an individual is applied to the procedure defined by the function-defining branch of its cooperation partner.

The variable `RS` is included in the function set of the result-producing branch. In this way, individuals in the initial population may contain the variable `RS` as internal nodes.

Also, a variable `EP`, called the *external procedure*, is defined for each individual to hold the current procedure to which the overall program is applied. Using this mechanism, the function-defining branch of one individual can be embedded within the structure of another individual. With this modifications, the application of the overall program to the `EP` procedure is caracterized by the following application:

```
((LAMBDA (RS)
   (PROGN
     (DEFUN ADF0 (Argument List)
       (VALUES (body of ADF0)))
   (VALUES
     (Body of Result Producing Brach)))) EP)
```

When creating each individual in the initial population, we initialize its corresponding `EP` variable so as to contain an arbitrary procedure with some fixed arity. This arity must be the same as the arity of the function-defining branch that will be shared, ensuring closure [9].

Cooperation Phase. In addition to the modifications to the program structure, we include a process for the logical progression of cooperation. After the execution of each evolution cycle, we execute a cooperation phase in which individuals exchange *procedures* to their mutual benefit. Each cooperation phase, in turn, consists of 4 stages: (1) a *selection stage* in which individuals choose potential cooperation partners; (2) an *evaluation stage* in which individuals estimate the payoff of cooperation; (3) a *decision stage* in which potential partners accept or refuse to cooperate; and (4)an *update stage* in which the structure of the programs are updated to include the modifications derived from the decision. This process will now be described in detail.

Selection Stage. For each individual in the population, a potential cooperation partner is selected using a tournament selection mechanism. Each individual choose a potential partner from a tournament of size n using fitness as the selection criterion.

Evaluation Stage. An individual then computes the potential outcome of cooperation. In our implementation, each individual shares its corresponding function-defining branch, which is bounded to the `EP` variable of the potential partner and viceversa. Each individual then recomputes its fitness through the application of the overall program to the `EP` variable. This modification to the program structure will generally result in an improvement or a degradation of the individual fitness.

Decision Stage. Individuals accept or refuse to cooperate using a decision procedure. We assume *global rationality* [8] for the decision procedure: each individual is interested in maximizing the overall outcome of the decision. The *overall outcome* is the sum of the outcome of both individuals. Under this assumption it is possible to emerge several ecological interactions, such as altruism and parasitism ([1]).

In this decision procedure, an individual may either accept (A) or refuse (R) to cooperate. We will use a matrix notation to represent the outcome of each action pair. We show an example in table 1.

The first individual selects one of the two rows by choosing A_1 or R_1; the second selects one of the columns by choosing A_2 or R_2. The individual outcome of cooperation corresponds to the numbers in the associated boxes: if, for example, actions A_1A_2 are selected, the row individual above would receive a fitness improvement of 3 while the column individual would receive a fitness improvement of 1. In this particular situation, each individual has a fitness improvement if the other individual accepts to cooperate. The situation above leads to mutual cooperation.

Table 1. Mutual cooperation

	A_2	R_2
	1	1
A_1	3	0
	0	0
R_1	3	0

Update Stage. Once the decision is made, the corresponding `EP` variables and fitness are updated, in order to execute the modifications derived from de decision.

4 Evolving Insect Locomotion with CGP

4.1 The Artificial Insect

The artificial insect we use is based on the model proposed by Beer [2]. The design of the body of the artificial insect is shown in figure 3.

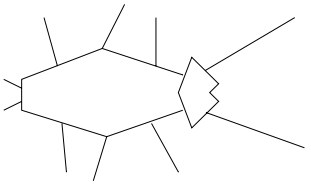


Fig. 3. Artificial insect [2]

The insect can raise and lower its legs, and is assumed that to be constantly moving. That is, when the insect raises one of its legs, a protraction movement is automatically activated. Similarly, when the insect lowers one of its legs, a retraction movement is automatically activated. Finally, both the return stroke and the power stroke are assumed to terminate in one time step.

The insect locomotion problem suggest a cooperative problem solving approach. In this work, we conceive locomotion as an emergent property of the cooperation between legs. Each leg may be viewed as an agent that cooperates with adjacent legs in order to move insect's body.

In particular, the problem of support is solved when a sufficient number of legs cooperate to remain the body of the insect stable. Similarly, the problem of progression is solved when a sufficient number of legs cooperate yielding local coordination and then making global coordination to emerge.

We propose two approaches to solve insect locomotion problem using cooperative genetic programming. The first is task-sharing and the second one is result-sharing. We also compare the performance of both methods.

4.2 Experiment 1: Task-Sharing

One of the key aspects of the task-sharing model is the existence of a decomposition of the overall problem. We decompose the locomotion problem into six subproblems. Each subproblem, in turn, consists of the local coordination between one leg and its adjacent legs. The adjacency relation is determined using the direction of the influences specified in the locomotion model proposed by Cruse.

A program in the population consists of six subprograms `PL1, PL2, PL3, PR1, PR2` and `PR3` each solving a local coordination problem. Additionally, a memory mechanism is asociated with each individual to represent the state (up or down) of each leg. We use a binary vector of six components for this purpose.

Terminal Set. To solve the coordination problem, a subprogram uses information related to its own state and to that of the adjacent legs at each time step. We define six 0-ary functions `SL1, SL2, SL3, SR1, SR2` and `SR3`. These functions return the state of a leg at a particular time step.

The terminal set for each subprogram consists of the following functions:

$$
\begin{aligned}
T_{\texttt{PL1}} &= \{\texttt{(SL1), (SL2), (SR1)}\} \\
T_{\texttt{PL2}} &= \{\texttt{(SL2), (SL1), (SL3), (SR2)}\} \\
T_{\texttt{PL3}} &= \{\texttt{(SL3), (SL2), (SR3)}\} \\
T_{\texttt{PR1}} &= \{\texttt{(SR1), (SR2), (SL1)}\} \\
T_{\texttt{PR2}} &= \{\texttt{(SR2), (SR1), (SR3), (SL2)}\} \\
T_{\texttt{PR3}} &= \{\texttt{(SR3), (SR2), (SL3)}\}
\end{aligned}
$$

Primitive Function Set. The primitive function set for all subprograms is:

$$F = \{\texttt{AND, OR, NOT, IF}\}$$

The set consists of the binary boolean functions `AND` and `OR`, the unary function `NOT`, and the 3-ary LISP function `IF`. This particular set satisfies the *closure* and *sufficiency* properties for this problem [9].

Fitness Function. Fitness is defined as the distance traveled by the insect as long as it remains stable. We assume that the insect advances one step if it is stable and at if at least one protraction movement is inmediatelly followed by a retraction movement.

Locomotion behavior is inherently periodic. This observation indicates that the execution of programs can reach an infinite loop. To avoid this problem, we define a maximum distance traveled by the insect. The genetic programming system terminates if a program reaches the maximum distance or if the system reaches the maximum number of generations.

Parameters of the Runs. We use a population size of 1000 individuals, and a maximum of 50 generations. Also, we set the maximum distance at 64 steps.

Results. We made several runs with the task-sharing model. In each run, we obtain individuals capable of sustained movement. The gaits obtained are similar to gaits observed in insects, including those derived from amputation. Figure 4 shows some of the gaits obtained with this experiment.

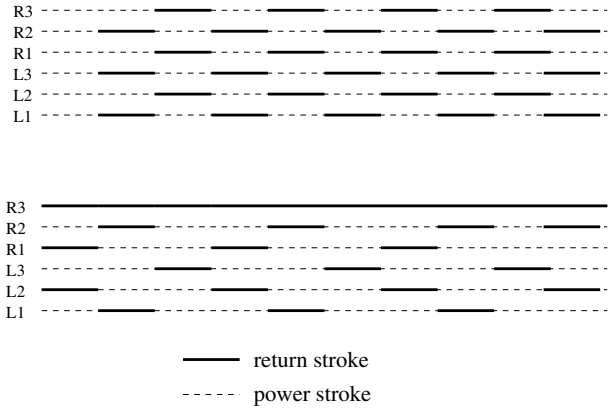


Fig. 4. Gaits obtained with task-sharing

4.3 Experiment 2: Result-Sharing

The result-sharing model of cooperative genetic programming incorporates automatically defined functions. We consider a program with one function-defininig branch and one result-producing branch. Aditionally, we associate a binary vector of six components to each program to represent the state of each leg.

Terminal Set. We define six 0-ary functions `SL1, SL2, SL3, SR1, SR2` and `SR3` as before for the terminals set. Each function returns the state of one leg at any time step.

The terminal sets for the function-defining branch and the result-producing branch are defined as follows:

$$T_{\texttt{ADF0}} = \{\texttt{ARG0, ARG1}\}$$
$$T_{\texttt{RPB}} = \{\texttt{(SL1), (SL2), (SL3), (SR1), (SR2), (SR3)}\}$$

Primitive Function Set. We define six unary functions `FL1, FL2, FL3, FR1, FR2` and `FR3` that produce side effects. These functions act as the identity function. In addition, each function updates a particular component of the state vector.

The set also includes the binary boolean functions `AND` and `OR`, the unary function `NOT`, and the 3-ary LISP function `IF`. The primitive function sets for the function-defining branch `ADF0` and the result-producing branch are:

$F_{\texttt{ADF0}} = \{\texttt{FL1, FL2, FL3, FR1, FR2, FR3, AND, OR, NOT, IF}\}$
$F_{\texttt{RPB}} = \{\texttt{RS, ADF0, AND, OR, NOT, IF}\}$

Fitness Function. As before, fitness is defined as the distance traveled by the insect as it remains stable. We assume that the insect advances one step if it is stable and if at least one protraction movement is inmediatelly followed by a retraction movement.

Parameters of the Runs. We use a population size of 1000 individuals, and a maximum of 50 generations. We set the maximum distance at 64 steps.

Results. We made several runs with the result-sharing model. In each run, we obtain individuals capable of sustained movement. Some of the gaits obtained are very similar to gaits obtained with the task-sharing model. Figure 5 shows some of the gaits generated with this experiment.

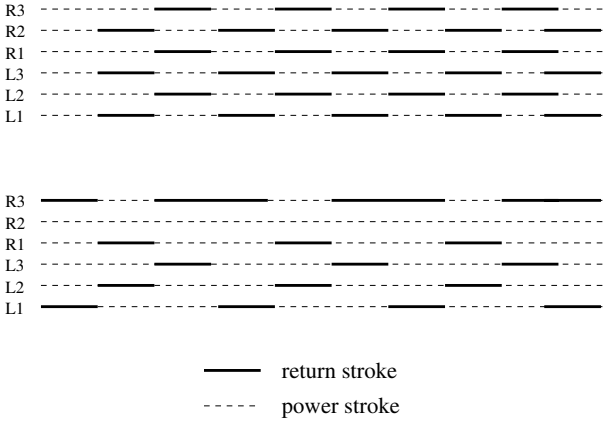


Fig. 5. Gaits obtained with result-sharing

4.4 Experiment 3: Comparative Analysis

The experiment consists of comparisons between the task-sharing model and the result-sharing model. Our results indicate that result-sharing outperforms task-sharing finding an insect capable of sustained movement.

We use the notion of *computational effort* $I(M, i, z)$ to analyse the performance of both methods ([9]). *Computational effort* is the number of individuals that must be processed to produce a solution by generation i with probability greater than z with a population of size M. Each cooperation phase counts as a generation in the result-sharing model.

For each problem, we perform 100 runs with task-sharing and 100 runs with result-sharing. We use a population size of 1000, and a maximum of 50 generations. The results are summarized in table 4.

Table 2. Results

Method	Computational Effort
Result-sharing	11,000
Task-sharing	13,000

5 Conclusions and Future Work

Cooperation is recognized as an important underlying principle of social systems. A natural extension of this principle is the application of cooperation to genetic programming systems. We showed that cooperative genetic programming can be used to evolve insect locomotion as an emergent behavior. In effect, locomotion emerges from the local cooperation between programs.

We introduce two models for cooperation in genetic programming systems based on task-sharing and result-sharing. The task-sharing model involves modifications to the program structure of conventional genetic programming and the design of restricted genetic operators. The result-sharing model involves modifications to the program structure of genetic programming with automatic definition of functions and a cooperation stage in which individuals share function defining branches for their mutual benefit. Further research will address cooperation in genetic programming by examining more test problems.

The focus of this study has been on the evolution of locomotion behavior and on the comparative performance of the proposed cooperation models. An immediate extension to this work is the addition of a blackboard to the result-sharing. This modification would accelerate the transmission of information between individuals. Other modifications include the consideration of a partner selection mechanism to identify the properties of individuals that have succesfully cooperated. The incorporation of a more complicated cooperation mechanism into genetic programming may result in the emergence of several complex ecological interaction patterns.

References

1. Axelrod, R. 1984. *The Evolution of Cooperation.* Basic Books, Inc.
2. Beer, R. 1990. *Intelligence as Adaptive Behavior.* Academic Press, Inc.
3. Brooks, R. 1989. A Robot that Walks: Emergent Behaviors from a Carefully Evolved Network. *Neural Computation.* **1:2**, pp 365-382.
4. Cobb, H. 1993. Is the Genetic Algorithm a Cooperative Learner?. In D. Whitley (ed.), *Foundations of Genetic Algorithms 2.* pp 277-296, Morgan Kaufmann Publishers, Inc.

5. Cruse, H. 1990. What Mechanisms Coordinate Leg Movement in Walking Arthropods. *Trends in Neurosciences*. **13**, pp 15-21.
6. Dawkins, R. 1976. *The Selfish Gene*. Oxford University Press.
7. Ferrell, C. 1995. A Comparision of Three Insect-Inspired Locomotion Controllers. *Robotics and Autonomous Systems*. **16:2-4**, pp 135-159. Elsevier.
8. Genesereth, M., Ginsberg, M., Rosenchein, J. 1988. Cooperation Without Communication. In A. Bond, L. Gasser (eds.) *Readings in Distributed Artificial Intelligence*. pp 220-226, Morgan Kaufmann Publishers, Inc.
9. Koza, J. 1994. *Genetic Programming II*. The MIT Press.
10. Luke, S., Spector, L. 1996. Evolving Team Work and Coordination with Genetic Programming. In J. Koza, D. Goldberg, D. Fogel, R. Riolo (eds.) *Genetic Programming 1996. Proceedings of the First Annual Conference*. pp 150-156, The MIT Press.
11. Smith, R. Davis, R. 1988. Frameworks for Cooperation in Distributed Problem Solving. In A. Bond, L. Gasser (eds.) *Readings in Distributed Artificial Intelligence*. pp 61-70, Morgan Kaufmann Publishers, Inc.
12. Spector, L., Luke, S. 1996. Cultural Transmission of Information in Genetic Programming. In J. Koza, D. Goldberg, D. Fogel, R. Riolo (eds.) *Genetic Programming 1996. Proceedings of the First Annual Conference*. pp 209-214, The MIT Press.
13. Spencer, G. 1994. Automatic Generation of Programs for Crawling and Walking. In K. Kinnear (ed.) *Advances in Genetic Programming*. pp 335-353, The MIT Press.
14. Vallejo, E. Ramos, F. 1999. Result-Sharing: A framework for Cooperation in Genetic Programming. In W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honovar, M. Jakiela, R. Smith (eds.) *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*. pp 1238, Morgan Kaufmann Publishers, Inc.
15. Wilson, D. 1966. Insect Walking. *Annual Review of Entomology* **11**, pp 103-122.

On the Minimal Cooperative Attitude in Multi-robotics Systems

Giovani Gómez and Fernando Ramos

Department of Computer Science. ITESM Campus Morelos.
Ap. Postal C-99 62050, Cuernavaca, Morelos. México.
Phone: +52(7)329-7143, Fax: +52(7)329-7141
{gegomez, framos}@campus.mor.itesm.mx

Abstract. A cooperative strategy for solving conflicting situations on mutirobotics motion planning is presented. The strategy is performed with a function with minimal sharing information, which controls that such movements really assists to both robots. We show that a simple function with cooperative sense can retrieve good paths in conflicting situations. As a rule, finest movements are neccessary to avoid collisions in hard environments such as inside a tunnel. We developed a novel technique inspirated in regularisation and Markovian processes. This technique is currently implemented in a randomized algorithm with a cooperative strategy for solving conflicting situations that often arrive in multi-robot manipulator environments. Randomized Algorithms (RA) have proven their capacity to escape from local minima. On the other hand, if collaborative attitudes are carried-out by the participants in the task, the time to solve conflicting situations can be considerably reduced. Moreover, when conflicting situations take long time to be solved, high risk of exploring not good free spaces can arrive and hence the search does not often drive directly the robots to their goals. Based on the above considerations, we built a framework of cooperative RA's aiming to facilitate the coordination of robots by establishing commitments to cooperate when conflicting situations appear, and by interacting through the exchange of data to go as directly as possible towards their respective goals.

1 Introduction

The basic problem of *classical* motion planning consists in the generation of a free collision path between an initial and a goal position for a robot in a static and completely known environment [13]. There is a strong evidence that a complete planner (i.e., one that finds an optimal path whenever one exists and indicates that no one exists otherwise) will take exponential time in the number of degrees of freedom (dof) of the robot. This type of problems are known as NP-Complete [5,3].

The computational complexity of the problem suggests the application of randomized techniques that make a tradeoff between completeness and speed of the algorithm. Generally speaking, the algorithms can be broadly classified in two: global and local planners. In the first case, a complete model is required

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 182–192, 2000.

of the free space, and in the second case the knowledge is bounded to a certain neighborhood of the robot's current position. In practice, the global approach is well suited to generate collision-free paths in the case of the basic problem. Such is the case of Probabilistic Roadmaps (PRM), a randomized method proposed by Kavraki and Latombe which has been successfully applied for solving motion planning problems in static and completely known environments [11].

Many applications in robotics need the participation of two or more robots working within the same workspace or sharing a common workspace. In most of the shared workspace, which is characterized by constantly changing environments and no predictable partially (dynamic domains), the action planning should be on line. The dynamic environments can bring about high risk of collisions between robots and the robots with the obstacles in the workspace.

Li [14] describes two approaches for path planning, centralized and decoupled. The centralized approach treats the various robots as a single robot, by considering the cartesian product of their individual C-spaces. This space is called composite C-Space. A drawback of this approach is that it leads to an exploration within a large-dimensional space, which may be too time-consuming. A composite C-Space computes many unnecessary places in the workspace. An alternative is a decoupled approach, which considers planning for one robot at a time. In a variant of this approach, the robots whose motions have already been planned are treated as moving obstacles, constraining the motions of the other robots. Another variant of this same approach, called velocity tuning, plans the path of each robot separately and then tunes the velocities of the robots along their respective paths so that no two robots ever collide. However, the decoupled approach is not complete, i.e., it could not find existing paths.

After having considered the characteristics of centralized and decoupled, let us remark that our approach is *decoupled* and planning can be done on-line.

2 Cooperation

Some applications in robotics involve two or more robots that collaborate by the participation of a group of mobile or manipulators robots for achieving a global task. For instance: the MARTHA project where a fleet of mobile robots develop transshipment tasks in harbors and airports [1]; an assembly application in [14] is carried out by two robots in a cooperative framework. We can say that in this type of collaborative tasks, the robots can act over the same object or not, whatever is the case, given that they share always the same work space and high risk of collisions are permanently present. One of the main research areas in RoboCup is the collaboration for developing and executing team strategies for solving conflicting situations under an optics of Multi-Agent Cooperative Systems [12].

The end effector and the joints of a manipulator are viewed as a set of agents that interact to move the arm until the end-effector has reached its objective [8].

In [2], a model based on information sharing among robots in multirobot systems is described. The information sharing deals with three aspects: task descriptions, adquisition of robot states, and adquisition of environment states.

The motion cooperation is treated through the use of information sharing about robot states and environment sensing.

The definition of cooperation in the present work is taken from distributed problem solving for cooperative agents systems: cooperation based on *shared tasks* and cooperation based on *shared results* [16]. The present work belongs to the last type of cooperation.

In the shared tasks, a set of subtasks, which are derived from the global task, are distributed to the agents and they cooperate to reach the global task. In the shared results, the agents share partial results in order to improve their solutions. If agents share their results in order to reach their goals, one robot could obstructs the path of other robot; in this case they do not have a "collaborative attitude". In fact, when robots perform a collaborative attitude, the path planning problem becoms easier.

In [15] Mazer et al., an algorithm which does not need a pre-processing step to find paths in high dimension spaces is presented. This method works for static as well as dynamic systems. The algorithm was tested with a 6 DOF robot that plans a path by considering that a second robot represents an obstacle in movement. This algorithm needs a sophisticated parallel architecture in order to show its advantages.

RA's, some times implemented as genetic algorithms, have been used in robotics. In [6] a cooperative genetic algorithm framework is based on shared data. In [17] Vallejo and Ramos have proposed a cooperative genetic programming algorithm for the coordination of insect leggs. This method can be extended for the coordination of multi-robot systems.

The match between MultiAgent Cooperative Systems (MACS) and MultiRobotic Systems (MRS) seems to be natural in such a way that robots can be considered as robotic-agents that should cooperate to achieve common goals. More precisely, we aim to build a framework of randomized collaborative algorithms based on exchanged data that aim at solving conflictive situations and reaching their respective goals as directly as possible without important deviations.

3 Scenario

In order to understand the kind of problems we want to solve in this work, an example of two robots whose initial position is located out of the tunnel and whose goal position is located at the opposite side inside the tunnel is shown in figure 1.

We can see that a risk of collision appears if two robots try to occupy the same free-spaces at the same moment, in particular, reduced space inside the tunnel (figure 1). In a collaborative framework, cooperative attitudes associated with robots can help to establish agreements between robotic-agents to decide about movements that can liberate spaces needed by each other in order to facilitate their motions. We illustrate this idea by the following example:

If *R1* obstructs the movements of *R2* to reach its goal, then *R2* can request *R1* to execute movements until a space is liberated for it; and, in this way, it can advance towards its goal and viceversa.

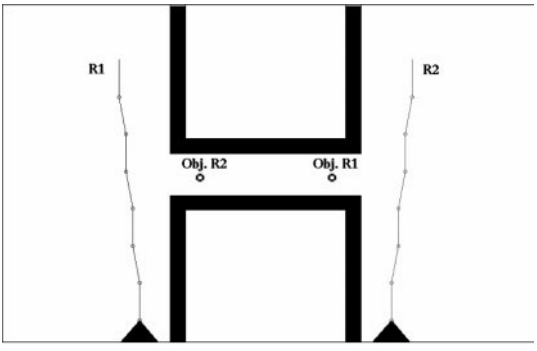


Fig. 1. Horizontal tunnel. The robot goals are at the opposite side.

As we can see, these collaborative attitudes can serve for solving conflicting situations such as risk of collisions inside the tunnel. These collaborative attitudes rely on a mechanism for data exchange between robots. This data is related to candidate movements that will be evaluated as possible collisions arise. A new compromise appears concerning the path smoothness described by the manipulators. That is, the reduced space inside the tunnel requires smooth paths to ensure that the segments of the robots do not collide between them.

A novel method based on regularisation techniques taken from digital image processing has been implemented for building smooth paths. Finally, we take advantage of one of the properties of RA's: their capacity to escape from local minima. Based on the considerations described above, we build a model that integrates two main aspects: RA's and cooperation based on shared results.

4 Randomized Cooperative Algorithm

This section presents the model proposed for the random algorithm with cooperative aspects.

Cooperation is established as a compromise that *should benefit both robots* in the search for their goals. Consequently, a kind of "Min-Max" algorithm is not applicable to our purposes. Consider, for example, a tunnel where both robots are inside as illustrated in figure 2.

If $R1$ executes an arbitrary movement, it could be an advantage for it, but the same movement can become an irreparable obstacle for $R2$. The *best movement* should be the *best one* for *both* robots. Therefore, each robot needs to be endowed with cooperative attitudes in the sense that they need to share their information in order to evaluate the consequences of a possible movement. A minimal cooperative function is defined, which returns two possibilities:

$$cooperative \quad message = \begin{cases} Obstruction \\ Good \quad movement \end{cases}$$

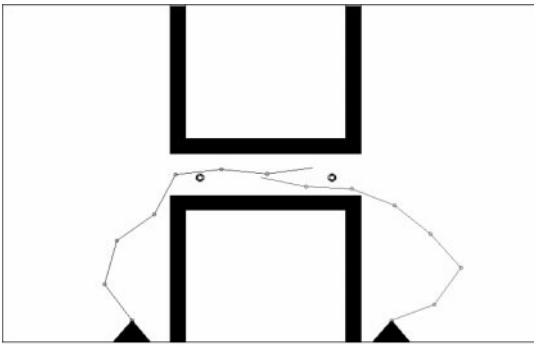


Fig. 2. Both robots inside a tunnel. Short movements are executed.

- *Obstruction:* One robot hides the *visibility* of its partner to advance to its goal.
- *Good movement:* Each attempting movement is evaluated as valid to go towards its goal.

Finally, for each attempt, we will use a kind of $Max - Max$ algorithm. In addition to cooperativeness, a new configuration, C_{new}, tries to minimise the distance between the current end effector position, ee_C, to the goal position, O. The metric used here, in 2D, is the euclidean distance between ee_C and its O.

$$distance(ee_C, O) = \sqrt{(ee_{C_x} - O_x)^2 + (ee_{C_y} - O_y)^2} \tag{1}$$

where ee_{C_x} y ee_{C_y} are:

$$ee_{C_x} = \sum_{i}^{dof} L_i cos\theta_i$$

$$ee_{C_y} = \sum_{i}^{dof} L_i sin\theta_i$$

and L_i is the length of each segment.

Summarising, the cooperative function can be expressed by combining the cooperative messages with equation (1). It evaluates the attempting position to reach the goal and the distance between their current positions to their goals. The idea is that a formal cooperative function *forces the cooperation* when a robot plans a movement to be executed. Let us explain this cooperative function with the following example: Let R be the robot that wants to move, and R_p the partner:

$$evaluation_R = distance(ee_R, O_R) + message(R, R_p) \tag{2}$$

where $evaluation_R$ means the evaluation associated with R. As we can see, expression (2) should capture, partially, the sense of a cooperative movement.

In order to ensure a better movement, R_p executes a short random-walk and returns its response to R. With this shared information, R can be sure that *its movement is not an obstacle* to R_p; in fact, the new position of R *really assists* R_p in its planning. Such considerations are written in the final minimal cooperative function:

$$\begin{aligned} evaluation = distance(ee_R, O_R) + distance(ee_{R_p}, O_{R_p}) \\ + message(R, R_p) + message(R_p, R) \end{aligned} \tag{3}$$

That is, the best movement must minimise the distance from ee_R and ee_{R_p} to their respective objectives O_R and O_{R_p}, allowing, at the same time, the visibility to their objectives.

However, if a priority is not associated with each robot for the execution of movements, a high risk of undecidable situations can arise. Let us review why with an example:

$R1$ sends a message to $R2$ indicating that it can execute a movement. At the same time, $R2$ sends a message to $R1$ indicating that it can execute a movement. And so on.

The precedent situation can be repeated infinitely. In order to avoid this kind of undecidable conflicting situations, we need to define a *priority* or *weight* in (3): if a cooperative situation is established, the robot having the highest priority could break the circular situation. For this reason, we need to associate with each message a priority coefficient defined by κ. The cooperative function (3) is transformed in the following expression:

$$\begin{aligned} evaluation = distance(ee_R, O_R) + distance(ee_{R_p}, O_{R_p}) \\ + \kappa_R message(R, R_p) + \kappa_{R_p} message(R_p, R) \end{aligned} \tag{4}$$

where κ_R and κ_{R_p} controls the relative priority (weight) of R with respect to R_p. If $\kappa_R = \kappa_{R_p}$ the system falls in an undecidable conflicting situation. In practice, it is not necessary to set two variables (κ_R and κ_{R_p}) because the relative weight can be normalised from cero to one, and only one variable would be set.

Before showing examples of expression (4), as mentioned above, it is neccessary to ensure that smooth movements are described by the end effector, in order to facilitate the motion of the robots. This will be described in the next section.

5 Smooth Movements

As a rule, fine movements are neccessary to avoid collisions in hard environments such as inside a tunnel. Robot movements should be smooth to get a better control when robots are moving around. We first introduce the concept of smoothness. Lets consider that a movement is executed from configuration C_n to C_{n+1}, which are represented as a set of joint values:

$$C_n = \{\theta_1, \theta_2, \theta_3, \ldots, \theta_n\}$$
$$C_{n+1} = \{\theta_1^{new}, \theta_2^{new}, \theta_3^{new}, \ldots, \theta_n^{new}\}$$

From this perspective, we expect to calculate each θ_i^{new} as closed as θ_i in order to ensure a smooth movement. Before discussing how a smooth movement can be formulated, let us review some useful considerations:

- The robot environment could be considerated as a 2D-pixel lattice (for planar robots).
- The 2D lattice is divided in a pre-defined neighborhood system (cliqué), normally of reduced few points.
- Each movement θ_i to θ_i^{new} *only* depends on its neighborhood, θ_i^{new} is calculated analysing only a local neighborhood.
- Also, θ_i^{new} must be as close as possible to θ_i except by $\vartheta{=}\Delta\theta{=}\mid \theta_i - \theta_i^{new} \mid$.

Note that the last definition of smoothness seems as a Markovian process running in a 2D lattice. We consider it as a Markovian process because of the following reasons:

- Let RxC be a lattice of R rows and C columns; and $N(r,c)$ a neighborhood. The neighborhood needs to satisfy that a point $(r,c) \in R$xC but $(r,c) \notin N(r,c)$. Because a markovian process in a 2D lattice is *stationary*, $(a,b) \in N(r,c)$ if and only if $(a+i, c+j) \in N(r+i, c+j)$ for any (i,j). This relation is true while computing θ^{new} in a robot, because the spatial neighborhood is the same all over the 2D lattice. In fact, all configurations would be calculated in a stationary way.
- The *conditional independence* assumption considers that the conditional probability of a (r,c) given all the remaining in the image is equal to the conditional probability of the point given just the points in its neighborhood. Formally,

$$P\{(r,c)|(i,j) : (i,j){\in}RxC, (i,j){\notin}(r,c)\} = P\{(r,c)|(i,j) : (i,j){\in}N(r,c)\}$$

  So that we can assume that the exploratory process for avoiding collision is equal to the locality given its neighborhood, in other words, it is a local process given the rest of the workspace.

Having shown some properties of Markovian processes applied to the robot motion, we now return to the smoothness calculus. Regularisation is a very useful technique to deal with Markovian processes, and it has been applied sucessfully in digital image processing under a so called Markov Random Field (MRF) [9][4]. The basic idea behind regularisation is that a system should *reach smoothly* its final configuration by defining a smoothing factor. It is neccessary to include a smooth factor because, usually, a MRF is an iterative process trying to minimise an energy function U. As an example we show the membrane model, also referenced as Ising model [10]:

$$U(f) = \underbrace{\sum_{r \in L} |p_r - \hat{p}_r|^2}_{likehood} + \underbrace{\lambda \sum_{<r,s>} |p_r - p_s|^2}_{smoothness}$$

In intuitive terms, and with no more details for our purpose, this thermodynamical model states that the solution, U, makes a tradeoff between the likehood of the observations (first term), and it should vary smoothly across the 2D lattice (second term); with the parameter λ controlling the relative weight of this smoothness constraint. Many people refer to first term as the *a priori* information or domain information. In our case, the domain information is *smooth movements*, as close as possible to the last robot configuration.

This model allows us to reconsider how to move an angle θ in our system. Then, clearly each θ in our robot needs, implicitly, a smoothing weight factor given by next relation:

$$\lambda_{base}\theta_{base}, \lambda_2\theta_2, \lambda_3\theta_3, \ldots, \lambda_{ee}\theta_{ee} \tag{5}$$

Obviously, if we wish a smooth behaviour of the end effector then the relation must satisfy:

$$\lambda_{base} < \lambda_2 < \lambda_3 < \ldots < \lambda_{ee} \tag{6}$$

Conceptually, λ_i is a *regularisation factor* associated with articulation i at any time. Regularising the angles represents an approach where articulations *closer to the end effector are privileged* in comparison to articulations closer to the base of the robot. We priorize them because a movement perform by θ_{base} could be inconvenient for getting smooth paths by the end effector.

Additionally, once a λ has been calculated we need to define the boundaries of θ_i in the following way:

$$\theta_i^{min} \leq \theta_i \leq \theta_i^{max} \tag{7}$$

We can now summarise the above considerations (5,6) in the following equation, which consists of a regularised approach to the robot movement:

$$C_{new} = \sum_{i}^{dof} \lambda_i\theta_i \tag{8}$$

which is restricted to the considerations given by (6,7).

6 Initial Results

Figure (3) shows the points traced by the end effector of both robots after executing functions (4) and (8). The cooperation can be illustrated by the following steps:

1. The robots execute actions “A” and “B”, figure (3), but do not try do enter into the tunnel because they become an obstacle to each other.
2. Both robots go down and again both robots do not enter into the tunnel. Figure 3, action “C” and “D”.
3. The right robot executes an up-movement, figure 3, action “E”, and in this way both robots can continue their planning.

Finally, we present three examples, figures 4, 5 and 6, with both robots reaching their final position. The cooperative algorithm was tested in a RS6000 H50, and it runs in a few seconds (less than one minute) from the initial to the final position.

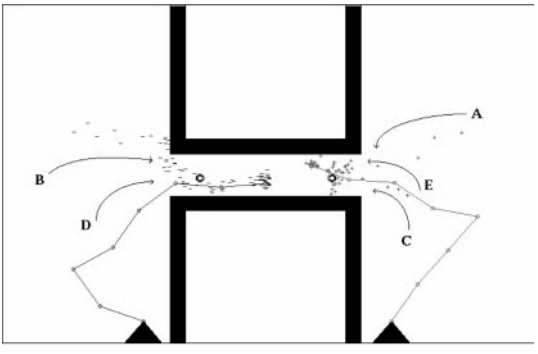


Fig. 3. A typical cooperative sequence: the right robot allows the entrance of left robot. Action "E".

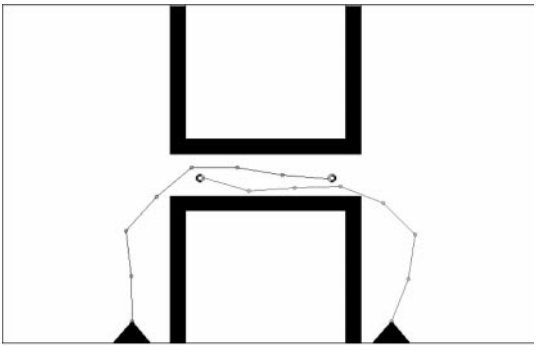


Fig. 4. Final state after executing the cooperative randomized algorithm. It shows a long hazardous trajectory.

Figures 4, 5 and 6 show hard environments where a *good* movement of the robots could be really an obstacle for other. From this point of view, the *minimal cooperative attitude* turns two isolated robots in really partners with common goals: reach their final position, and assist the other partner; obviously, at the same time.

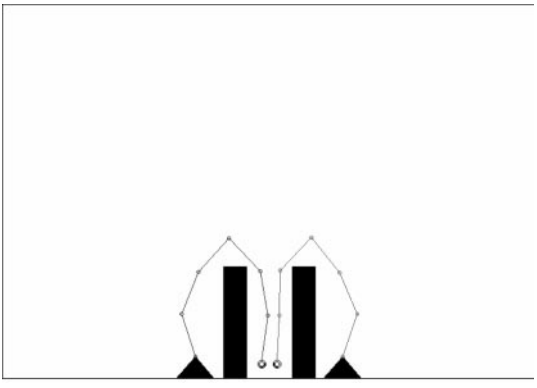


Fig. 5. Final state after executing the cooperative randomized algorithm in another kind of tunnel. At the same time, both robots needs to move slowly inside.

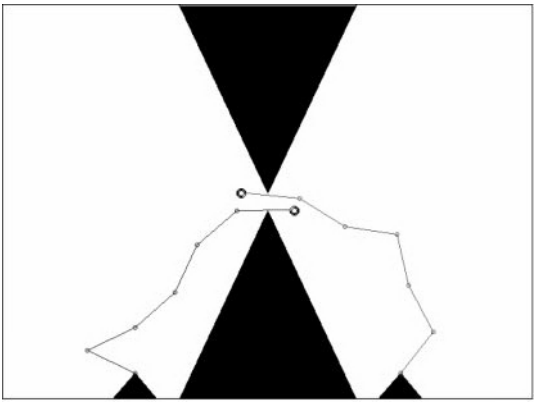


Fig. 6. Final state after executing the cooperative randomized algorithm in another kind of tunnel. Although the conflictive region is reduced, there are only one hazardous solution with both in their final state.

7 Conclusions and Future Work

We have presented a minimal and fast function with cooperative attitudes. Cooperative attitudes are associated with robots facilitating the solution of conflictive situations. Simple cooperative operators, that contain results to be shared by both robots, are able to liberate spaces in order to allow movements as directly as possible towards their goals. Note that a cooperative function with minimal sharing information is presented, which controls that the *movement should benefit both robots*. We explore the strength of MACS concepts related to cooperation in MRS, contributing with an alternative method for solving the cooperative motion planning problem on-fly. In addition, a novel and successful approach for getting smooth paths based on a regularisation technique has been developed.

Future research work will encompass developing more cooperative operators and generalise the technique to handle more than two robots in a shared workspaces. Finally, we aim to test the method in a real environment.

Acknowledgements. The authors are grateful to the anonymous reviewers and Dr L.E. Sucar for their many useful suggestions that helped us improve the final draft.

References

1. R. Alami, S. Fleury, M. Herrb, F. Ingrand, F. Robert: Multi-Robot Cooperation in MARTHA Project. IEEE Robotics and Automation. **5(1)**, pp. 36-47, 1998.
2. A. Cai, T. Fukuda, F. Arai: Cooperation of multiple robots in cellular robotic system based on information sharing. IEEE Int. Conf. on Advanced Intelligent Mechatronics, **1**, pp. 20, 1997.
3. J. Canny: The Complexity of Robot Motion Planning. MIT Press, 1988.
4. R. Chellappa, A. Jain: Markov Random Fields: Theory and Applications. Academic Press, 1993.
5. S. A. Cook: The complexity of theorem-proving procedures. Proceedings of the 2rd Annual ACM Symposium on the Theory of Computing, pp. 151-158, 1971.
6. V. de la Cueva, F. Ramos: Cooperative Genetic Algorithm: a new approach to solve the path planning problem for Cooperative Robotic Manipulators sharing the same work space. Proc. of IROS, **1**, pp. 267-272, 1998.
7. J. Desai, C. Wang, M. Zefran, V. Kumar: Motion Planning for Multiple Mobile Manipulators. Proc. of ICRA, **3**, pp. 2073-2078, 1996.
8. J. Ferber: Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Longman: Addison Wesley, 1999.
9. S. Geman, D. Geman: Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, IEEE Trans. on PAMI, **6(6)**, pp. 721-741, 1984.
10. E. Ising: Zeitschrift für Physik, **31**, pp. 253, 1925.
11. L. Kavraki, J. C. Latombe: Probabilistic Roadmaps for Robot Path Planning. In Practical Motion Planning in Robotics: Current and Future Directions, K. Gupta and A. del Pobil Eds., Addison-Wesley, 1998.
12. H. Kitano, M. Asada, I. Noda, H. Matsubara: Robocup: Robot World Cup. IEEE Trans. on Robotics and Automation, **5(3)**, pp. 30-36, 1998.
13. J.C. Latombe: Robot Motion Planning. Kluwer Academic Press, 1991.
14. T. Li, J.C. Latombe: On-line Manipulation Planning for Two Robot Arms in a Dynamic Environment. Technical Report, Robotics Lab., Department of Comp. Sci., Stanford University, 1997.
15. E. Mazer , J.M. Ahuactzin, D. Bessiere: The Ariadne Clew's Algorithm. Journal of Artificial Intelligence Research. **9** (1998), pp. 295-316.
16. R.G. Smith, R. Davis: Frameworks for cooperation in distributed problem solving. A. Bond, L. Gasser, eds. Reading in Distributed Artificial Intelligence. Morgan Kaufmann, 1991.
17. E. Vallejo, F. Ramos: Result-Sharing: A framework for cooperation in genetic programming. Proc. of GECCO, june 1999.

Eigenfaces Versus Eigeneyes: First Steps Toward Performance Assessment of Representations for Face Recognition

Teófilo Emídio de Campos, Rogério Schmidt Feris, and Roberto Marcondes Cesar Junior

CreatiVision - Creativision Research Group
Department of Computer Science
DCC-IME-USP, University of São Paulo
Rua do Matão, 1010, São Paulo, SP, 05508-900, Brazil
Phone: +55 (0) 11 818 6235, Fax: +55 (0) 11 818 6134
{teo, rferis, cesar}@ime.usp.br
http://www.ime.usp.br/~cesar/creativision

Abstract. The Principal Components Analysis (PCA) is one of the most successfull techniques that have been used to recognize faces in images. This technique consists of extracting the eigenvectors and eigenvalues of an image from a covariance matrix, which is constructed from an image database. These eigenvectors and eigenvalues are used for image classification, obtaining nice results as far as face recognition is concerned. However, the high computational cost is a major problem of this technique, mainly when real-time applications are involved. There are some evidences that the performance of a PCA-based system that uses only the region around the eyes as input is very close to a system that uses the whole face. In this case, it is possible to implement faster PCA-based face recognition systems, because only a small region of the image is considered. This paper reports some results that corroborate this thesis, which have been obtained within the context of an ongoing project for the development of a performance assessment framework for face recognition systems. The results of two PCA-based recognition experiments are reported: the first one considers a more complete face region (from the eyebrows to the chin), while the second is a sub-region of the first, containing only the eyes. The main contributions of the present paper are the description of the performance assessment framework (which is still under development), the results of the two experiments and a discussion of some possible reasons for them.

1 Introduction

Research on automatic recognition of faces is relatively recent, but it has been addressed by a many scientists from several different areas. According to Chellapa [1], several methods have been proposed, such as statistical-based, neural networks and feature-based. Currently, one of the methods that yields one of the most promising results on frontal face recognition is the Principal Component Analysis (PCA), which is a statistical approach where face images are expressed as a subset of their eigenvectors, hence called *eigenfaces*. This representation is used together with some

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 193—201, 2000.

classification technique for face recognition, e.g. a neural network. Next section discusses with more detail this technique.

Despite the nice results that can be obtained, this technique has the disadvantage of being computationally expensive because all pixels in the image are necessary to obtain the representation used to match the input image with all others in the database. This paper presents the experiments and the results of an approach that aims at reducing the computational effort of this approach. This technique is discussed below.

Some researchers have used eigenfaces and other eigenfeatures in order to perform recognition. The term *eigenfeature* has been used by Baback in [2], referring to the application of PCA in restricted areas of the image in order to obtain the main components of feature points of the face, such as the mouth (eigenmouth), the nose (eigennose), and the eyes (eigeneyes). In this sense, Brunelli's work [3] presents some interesting results. The results reported in that work obtained using a template covering only the eyes region are surprisingly better than the results obtained using a template that covered the whole face. Baback [2] has also obtained better results with eigenfeatures that included the eyes, the nose and the mouth than with eigenfaces.

The experiments reported here belong to a broader project aiming at the establishment of a performance assessment framework for face recognition problems. In this context, the results presented here have been obtained to either confirm or reject the results achieved by Brunelli in a PCA-based system, but using a different image database and specific preprocessing. The difference between this work and Baback's is that the present experiments only consider the eyes. As it will be seen, the present results corroborate those works, thus paving the way for the implementation of PCA-based face recognition systems that are faster and more efficient, due to the fact that they consider a smaller window.

The next section describes a PCA-based recognition system. We then describe the face image database used and the generation of the data for the training the classifier and testing. Section 4 shows the obtained results. We then move on to discuss the results and future work. The last section presents this work's conclusions.

2 Methodology

The experiments require that the sub-images (face and eyes region) be extracted from the original images. In order to improve PCA classification, the segmented images are normalized so that the face and the eyes images are of the same size. The obtained images are then used to train the PCA system, and to perform the classification tests. This section describes these procedures, namely the Principal Components Analysis system, and the image database formation.

2.1 Image Database

The image database adopted is composed of sixteen images of adult peoples; lots of them wear glasses, moustache and/or a beard. There are also major variations in their hair lengths. Most men are white, but there are also other ethnic groups present.

Moreover, the face images may vary with respect to illumination, face expressions and pose. Nevertheless, in all the considered images the two eyes are visible, i.e. There is no image with self-occlusion problems, as it would be the case for profile images.

The database has six different images for each person. Two tests have been carried out, the first one had been made using three images to train the system, and one test image; and another one using five images in the set of training for each person and one for testing. Tests with images belonging to the training set have also been performed.

The training set has been chosen randomly from the available images of each person. If we choose only "nice" images for the training set, the performance of the recognition system would decrease [4], because if that was the case, then the classification algorithm would have more difficulties in classifying faces with different pose, facial expression or different illumination conditions.

In the case of the tests with the region of the eyes, the images are generated from the original database. Such images are hand-cropped from the original face images so that only the region from the eyebrows down to a little below of the eyes are taken into account. On the other hand, in the full-face images, only the region that encloses from the forehead until the chin is actually used and, therefore, the hair is not considered in these tests.

As it has been commented above, both images sets (eyes and faces) have been hand-cropped. An alternative to this approach would be to adopt automatic methods for detecting feature points in faces (such as [7, 8 and 9]). Nevertheless, there is no well-established, general and really reliable algorithm for performing this task, which has motivated the use of hand-cropped images.

Fig. 1. Example of images used in the training phase.

Fig. 2. Example of test image.

Therefore, the image database is composed of 192 images of 16 people, being 96 images of faces and 96 of eyes and each person being represented by four different images of eyes and face. The systems are trained independently with respect to the eyes and the faces.

The original image resolution is 512 x 342 pixels with 256 gray levels. After the segmentation and normalization (described in section 2.3), both the images of eyes and of faces have been represented with resolution of 64 x 64, with 256 gray levels.

The figure 1 shows some examples of images from the database. Figure 1 shows the images used in the training set while figure 2 presents the test image.

2.2 Segmentation and Normalization

As it has already been commented, each image has been hand-cropped in order to generate two sub-images: *img1*, corresponding to the eyes region, and *img2*, encompassing the eyes, nose, mouth and part of the chin. An interactive program using specially developed GUI has been implemented. In this program, the human operator firstly clicks on the center of the left and of the right iris. From the obtained iris coordinates, the image is rotated so that the line between the two points become horizontally oriented. Next, the sub-images *img1* and *img2* are automatically cropped based on the distance between the eyes (i.e. the distance between the marked iris centered points). Therefore, let d be the distance between the clicked points. *img1* (the eyes image) is of size $0.65d$ x $1.8d$ pixels, with the clicked points being located at line $0.4d$, which implies that *img1*encloses a larger region above the in-between iris line, including the eyebrows, as desired. *img2* is obtained in an analogous way, except that it has $2.15d$ rows. These proportions have been found empirically after some experiments with the original face database.

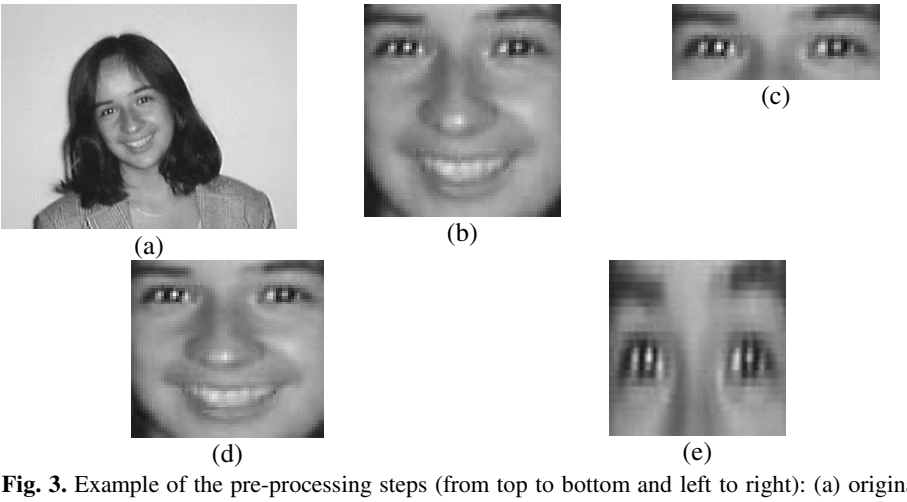


Fig. 3. Example of the pre-processing steps (from top to bottom and left to right): (a) original image (512x342); (b) result of the rotation and the segmentation of the face region; (c) resulting eyes region; (d) resizing of the face image; (e) resizing of the eyes region (both with 64x64 pixels).

Finally, because the Principal Components Analysis involves some multiplication of arrays, it is important that normalize the size of all images. This is done by resizing all images to 64 x 64 pixels. This final image size has been chosen because of the trade-off between computational cost and minimum resolution in order to guarantee that information about eyes, nose and mouth is not lost in too small image versions. Figure 3 shows the results of the segmentation and normalization processes for an image from the database.

2.3 Obtaining the Feature Spaces for Eyes and for Faces

In the first step of PCA, each 2D image is transformed in a 1D vector by appending the second image line after the first, followed by the third line, the fourth, and so on. The length of the obtained vector is $w\ h$, where w and h are the number of columns and rows of the input image, respectively (recall that, in the case of the experiments of this paper, $w = h = 64$). Therefore, each image is represented as a vector of a $(w\ h)$-dimensional space.

In the present case, when a set of face images are expressed in the above vector-like representation, the corresponding vectors tend to form clusters in the vector space, because of the common visual features of these images (eyes, nose, mouth,...). PCA allows the representation of each face (a point in the aforementioned vector space) by a few components that are necessary to distinguish between faces of different people. A database is formed from the training set of each person, and recognition is achieved by assigning the input face (to be recognized) to a specific class (person) by proximity of these representation components. This idea is applied both for the eyes-based and the face-based recognition.

2.4 The Principal Components Analysis System

To compute the principal components of faces or eyes we must do the following steps. First, we have to get the L first principal components of the image database. The principal components considered are the eigenvectors of the covariance matrix W. This matrix are taken from this product: $w = h \times X^t$, where X is the array constructed from the database, in with each column of X is a vector of an image described on the previous sub-section, and X^t is the transposition of the matrix X.

In this context, the first eigenvector of W is oriented in the direction of the largest variance among the faces, and it corresponds to an "average face" or an "average eye", because this vector have features shared to all the pictures. Therefore, it seems like a blurred face image. This blurring effect occurs because of the variations between the pictures. The second eigenvector of W characterizes the face features that are different from the first eigenvector, the third characterizes the points that are different from the two others eigenvectors, and so on. The eigenvectors are orthogonal to each other. It is important to note that the eigenvectors are sorted according to their respective eigenvalues, that is, the first eigenvector has the largest eigenvalue. The number of eigenvectors used by the classifier is fundamental the

recognition rate and execution time of the sorting. The recognition rate was tested varying the number of eigenvectors and the results are described later.

A PCA-based system can be implemented by using a self-associative memory or a statistical classifier. In the case of the face recognition system by a neural network, each element of the described vector in the previous sub-section is input for the classifier and each neuron has an output that contains a reconstructed pattern. The weights of this network are gotten from the matrix *W*.

In the statistical classifier approach, *W* it is treated as a covariance matrix that is used to create a spatial basis where the covariance among its elements is minimal. In this sense, it is possible to obtain reduction of the faces or eyes basis. Therefore, we can recover an image through an input image by doing some matrix computations. The computational cost of this technique is less than the neural network approach, but it has a larger error rate. More details about PCA can be found in Romdhani [4] and Valentin [5]. The figure below shows some (four) of the eigenvectors of the image database, that has 150 images from 15 persons (10 images per person).

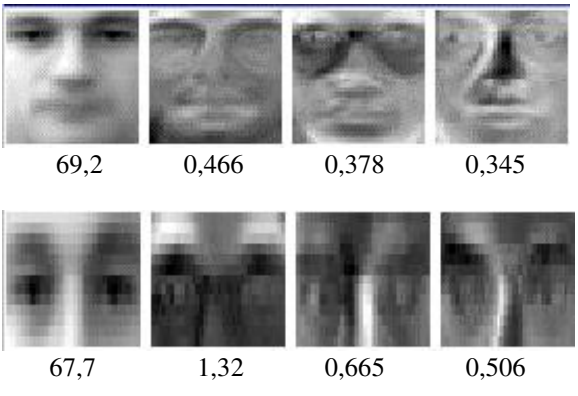


Fig. 4. Images of the first four eigenvectors with their respective eigenvalues of the faces database (above) and eyes database (below), which have been created using 150 images from 15 people (10 images per person: 5 face images and 5 eye images).

Figure 5 (in the next page) shows some examples of faces reconstructed thru the matrix *W*, that was obtained from the training image base described above.

2.5 Tests

As mentioned in section 1, the aim of the tests is to obtain a comparative study between person recognition using images that only contain the region of the eyes and images with the whole face.

Both tests have been done using the same recognition system based on PCA, the same image database, and the training and test sets corresponding to the same original images. The recognition rate of the system is adopted as the comparative criterion to analyze the results. Next section presents a discussion of our preliminary results.

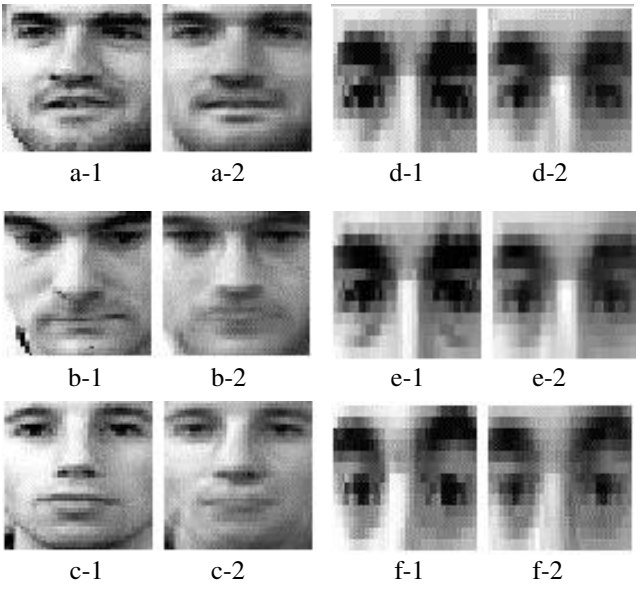


Fig. 5. a, b and c: face images; d, e and f: respective eye images. a-1, b-1, c-1, d-1, e-1 and f-1: original images; a-2, b-2, c-2, d-2, e-2 and f-2; reconstructed images. The reconstruction was done trhu the covariance matrix of a PCA system trained with 15 persons, 6 images per person.

3 Preliminary Results

As it has been noted before, the present work presents the current (preliminary) results of an ongoing performance assessment project. Section 4 presents a description of the future developmetns of the project.

Table 1 contains the results obtained by the experiments done with 16 people images. In this test set, we have used 3 images per person in the training set and 1 image per person in the test set. The results shown below was taken with training and test sets without intersection.

Table 1. Recognition rate (%) of the PCA system using eyes and face. The training was done with 3 images per person.

Number of Eigenvectors	Eyes	Faces
3	25,00	31,25
4	25,00	37,50
5	50,00	37,50
10	56,25	37,50
13	62,50	43,75
15	62,50	43,75
24	62,50	43,75
48	62,50	43,75

The above results show a poor performance because of the limited number of samples of the training set for each person. In order to improve the results, some experiments using a larger training set have been carried out, and Table 2 shows some results for this case. Here we used images from 15 people, 5 images per person in the training set and 1 image per person in the test sets.

Table 2. Recognition rate (%) of the PCA system using eyes and face. The training was carried out with 5 images per person.

Number of Eigenvectors	Eyes	Faces
3	40,00	46,67
15	73,33	66,66

Note that, if the classifier uses more than four eigenvectors, the performance of the eye recognition system is superior to the face recognition system. Moreover, the recognition rate increases significantly for both classifiers when the training set size is increased. The better performance for the eyes based classifier can be explained by two main reasons. Firstly, the inclusion of the nose and mouth region can reduce the recognition performance because face expressions implies strong distortions in this region. Furthermore, the complexity of a recognition system increases with the number of used features. This fact implies that the number of objects required to train the classifier and measure its performance increases exponentially with the number of characteristics. Therefore, adding characteristics that are either noisy or highly correlated to each other may decrease the classifier's performance if a large enough training set is not available. This fact is well known in the pattern recognition research area and can be found in further detail in [6,10,11 and 12]. Despite the lack of a larger set of classes (people) used as input and of tests with different training set sizes, the results obtained so far corroborates this theory.

4 Conclusions

This paper describes some results of a PCA-based recognition's technique applied to people recognition within the context of performance assessment. Tests with eigenfaces and eigeneyes were performed, and it was found that in most cases eigeneyes provide a superior recognition performance than eigenfaces.

In spite of eigeneyes have less information than eigenfaces, these obtained results are understandable because an increasing number of features also increases the complexity of the system. Although only a limited number of tests have been performed, the results show that images which contain only eyes are sufficient to obtain good results in face recognition. In fact, eyes differ considerably from person to person.

Observing the results, we conclude that faster face recognition systems based on PCA can be implemented by using eigeneyes instead of eigenfaces. It is easy to realize that, in the pre-processing step, less computational effort can be required using only eye images, since the images can be smaller. Therefore, also the required training set is smaller when we use eigeneyes.

We are currently working to perform more tests considering variations in the number of eigenvalues and variations in the training set size. Furthermore, the number of people to be recognized will be increased, using face databases with variations in pose, facial expression and illumination. Therefore, more reliable results will be obtained. It is important to note that all these issues of representation for recognition are central to the ongoing research on face recognition by the vision community [12].

Acknowledgments. Roberto M. Cesar Jr. is grateful to FAPESP for the financial support (98/07722-0), as well as to CNPq (300722/98-2). Rogério Feris and Teófilo Campos are grateful to FAPESP (99/01487-1 and 99/01488-8).

We are grateful to Carlos Hitoshi Morimoto (IME-USP) for useful discussions.

References

[1] R. Chellappa, C. L. Wilson and S Sirohey, "Human and machine recognition of faces: a survey", *Proceedings of the IEEE*, vol. 83, no. 5, may 1995, pp 705-640.

[2] B. Moghaddam and A. Pentland, "Face Recognition using View-Based and Modular Eigenspaces", *In: Proc. of Automatic Systems for the Identification and Inspection of Humans*, SPIE vol. 2277, July 1994.

[3] R. Brunelli and T. Poggio, "Face recognition: features versus templates", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, no. 10, pp. 1042-1052. October 1993.

[4] S. Romdhani, "Face recognition using principal conponents analysis", *MSc Thesis, Department of Electronics and Electrical Engineering, University of Glasgow*, March 13th 1996.

[5] D. Valentin, H. Abdi, A. J. O'Toole (in press)., "Principal component and neural network analyses of face images: Exploratinos into the nature of information available for classifying faces by Sex.", In C. Dowling, F.S. Roberts, P. Theuns, *Progress in Mathematical Psychology*. Hillsdale: Erlbaum.

[6] K.R. Castleman, "Digital Image Processing", *Prentice-Hall*, Englewood Cliffs, NJ, 1996.

[7] Silva, L., Aizawa, K., Hatori, M., "Detection and tracking of facial features." *In Proc. of SPIE Visual Communications and Image Processing'95* (VCIP'95), pp.2501/1161-2501/1172, Taipei, Taiwan, May. 1995. Awarded the SPIE (The International Society for Optical Engineering) Best student paper.

[8] Rowley, H.A., Baluja, S., Kanade, T., "Neural Network based Face Detection.", *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp:23-38, January 1998.

[9] M. La Cascia e S. Sclaroff, "Fast, Reliable Head Tracking under Varying Illumination", *Technical Report, Boston University* BU CS TR98-018. To appear in Proc. IEEE CVPR99.

[10] A. Jain and D. Zongker, "Feature Selection: Avaluation, Application, and Samall sample Performance", *in IEEE Transacrions on Pattern Analysis and Machine Inteligence*, pp. 153-158 vol. 19, no. 2, February 1997.

[11] L. I. Perlovsky, "Conundrum of Combinatorial Complexity", *IEEE Transacrions on Pattern Analysis and Machine Inteligence*, pp. 666-670 vol. 20, no. 6, June 1998.

[12] Craw, N. Costen, T. Kato, and S. Akamatsu , "How Should We Represent Faces for Automatic Recognition?", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):725-736., 1999

A Neurocontrol Scheme of a 2-DOF Manipulator Using CMAC

Raúl Leal Ascencio[1*] and Marco Perez Cisneros[2]

[1] Depto. de Electrónica, Sistemas e Informática
Instituto Tecnológico de Estudios Superiores de Occidente, ITESO,
Tlaquepaque Jal., 45090, México
rleal@iteso.mx

[2] Depto. de Electrónica, Sistemas e Informática
Instituto Tecnológico de Estudios Superiores de Occidente, ITESO,
Tlaquepaque Jal., 45090, México
marcop@cucs.udg.mx

Abstract. Artificial Neural Networks (ANN) are an emerging technology, yet, in continuous dynamic behavior, much work has been done to attempt to generate a formal method to design a controller based on this technology. Based on this fact we are starting to work towards contributing to the generation of a formal method of the application of neurocontrol and thus present several control schemes using multilayer perceptrons and also Albus' Cerebellar Model Articulation Controller (CMAC). The control schemes are presented and explained. The direct inverse neurocontroller model is used as the system controller without a linear controller. Results from two neurocontrollers are presented in a comparative way versus linear system using a 2-DOF plant.

1 Introduction

Artificial Neural Networks (ANN) are an emerging technology, yet, in continuous dynamic behavior, much work has been done to attempt to generate a formal method to design a controller based on this technology. Based on this fact we are starting to work towards contributing to the generation of this formal method of the application of neurocontrol and thus present several control schemes using multilayer perceptrons and also Albus' Cerebellar Model Articulation Controller (CMAC) [1].

ANN applications in control are characterized by operating in an ill-defined, time-varying environment, adapting to changes in the plant's dynamics as well as the environmental effects, learning significant information in a stable manner and placing few restrictions on the plant's behavior. It is hoped that increased adaptation will result in improved system performance, increasing the quality of the solution and reducing the design cost. Neural algorithms have much to offer to the control engineer given the above characteristics [4].

* Corresponding author

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 202—211, 2000.

A key feature of neurocontrol is that the non-linear activation functions of neurons lend themselves naturally to the control of systems whose dynamics are highly nonlinear and unknown or uncertain. The inverted pendulum and double inverted pendulum systems are bechmark nonlinear dynamic systems which are inherently unstable and thus attractive for testing control schemes. One objective of this project is to investigate the use of the neural network schemes for the control of the simple inverted pendulum system for the preliminary experiments and a double inverted pendulum system as the final objective.

2 Description of the Control Scheme

Towards fulfilling the above objective, system identification for the simple pendulum and double pendulum models are first carried out. A neurocontroller (fig. 1) is then designed using neural network model for the pendulum. We show that the controller is found to be stable in the presence of sensor and actuator noise and parametric uncertainty (i.e. modeling errors).

The simple inverted pendulum neurocontroller is trained with a linear controller made via the 'Pole Placement Design' method. A direct inverse control scheme is used as the control element [5][6][7]. Fig. 1 shows the neurocontrol block diagram using a direct inverse scheme. The input to the model receives signals from states of the plant and its output is a control signal directed towards the inverted pendulum. The direct inverse model is used as the system controller without a linear controller [3].

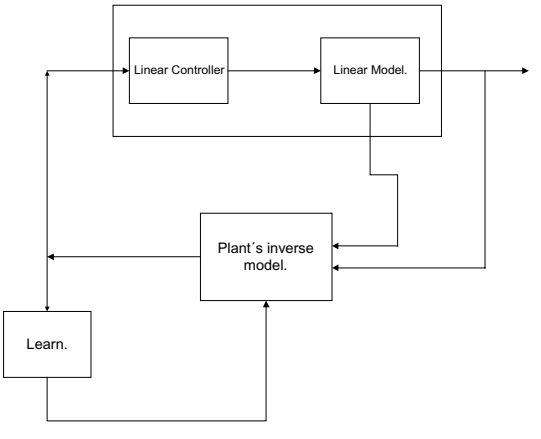


Fig. 1. Direct inverse-model of the plant.

The double inverted pendulum is modeled through the dynamics of the plant using a direct inverse model. This neural model will be used as a reference model in a direct adaptive control scheme [2]. This scheme is presented in fig. 2 and it serves to train a neurocontroller. In this scheme, the target output is defined through the output of the reference model, which allows adjusting the control signal in a stable manner so that the plant´s output asymptotically tracks the output of the reference model [4]. The

performance of this algorithm depends on the choice of a suitable reference model and the derivation of an appropriate learning mechanism.

Feedforward networks have been used with the Levenberg-Marquardt optimization algorithm for training for both system identification and controller training phases which results in superior learning as compared to regular backpropagation or backpropagation with momentum and adaptive learning rate. On implementing the CMAC technique, one appealing feature is its efficient realization in software in terms of training time and real-time operation. Modelling and training of CMAC are presented also through its application to the simulations of the plant.

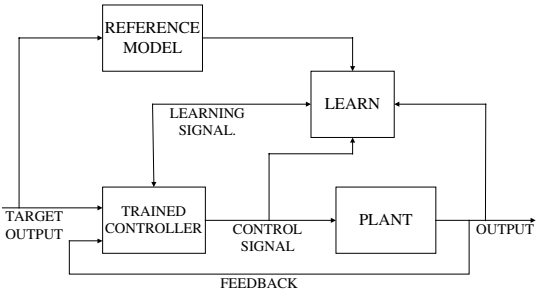


Fig. 2. Model reference control arquitecture.

3 The Plant's Model

The inverted pendulum system is a typical benchmark dynamic non-linear system. Two such systems are shown on figure 3. For this initial experiments, the simple inverted pendulum (fig. 3a) is being used as the plant and we will move on to control the double-inverted pendulum in the as a next step. A state-space approach is used to generate a plant model. Then a controller is designed using the pole placement design technique. The design is formulated in terms of obtaining a closed-loop system with specific pole locations. The controller generates a control signal that is going to be applied to the inverted pendulum in order to control the arm in a vertical position.

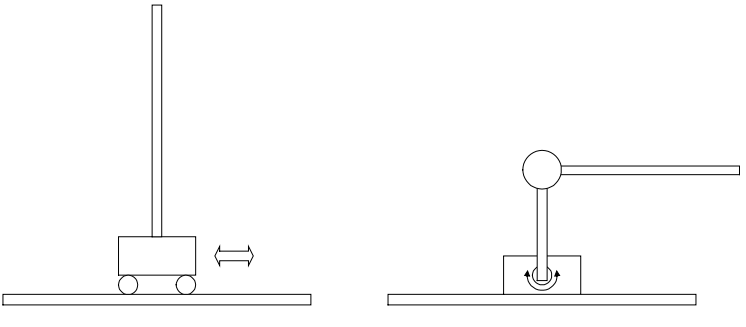


Fig. 3. Two possible plants. a) the simple pendulum. b) the double inverted pendulum.

The inverted pendulum's model expressions are:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{M+m}{Ml}g & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{m}{M}g & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{1}{Ml} \\ 0 \\ \frac{1}{M} \end{bmatrix} u \qquad 1)$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \qquad 2)$$

Using an Ackerman´s solution for pole placement, with: μ_1=-2+j3.464, μ_2=-2-j3.464, μ_3=-10, μ_4=10, we find a controlled system expression:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -277.549 & -60.997 & -163.099 & -73.394 \\ 0 & 0 & 0 & 1 \\ 148.5845 & 30.3485 & 81.5495 & 36.697 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \qquad 3)$$

with a control force equal to:

$$u = -Kx = 298.15\, x_1 + 60.697\, x_2 + 163.099\, x_3 + 73.394\, x_4 \qquad 4)$$

3.1 The Second Plant

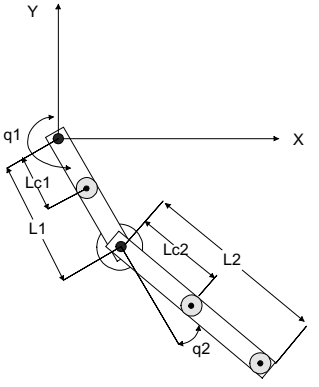
In this paper we describe the Pendubot™[9], a mechatronic device for use in control engineering education and for research in non-linear control and robotics. This device is a two-link planar robot with an actuator at the shoulder but no actuator at the elbow. With this system, a number of fundamental concepts in non-linear dynamics and control theory may be illustrated. The pendubot complements previous mechatronic systems, such as the inverted pendulum. We will discuss the design, and control of the Pendubot™ , a two-link, underactuated robotic mechanism that we are using for research in non-linear control and to teach students in various concepts in non-linear dynamics, robotics, and control system design. The Pendubot™ consists of two rigid aluminium links of lengths 9 in and 6 in, respectively. Link 1 is directly coupled to the shaft of a 90V permanent magnet DC motor mounted to the end of a table. The motor mount and bearings are then the support for the entire system. Link 1 also includes the bearing housing for joint two. Needle roller bearings riding on a ground shaft were used to construct the revolving joint for link 2. The shaft extends out both directions of the housing allowing coupling to the second link and to an optical encoder mounted on link one. The design gives both links full 360º of rotational motion. Link 2 is

constructed of a 1/4 inch (0.635 cm) thick length of aluminium with a coupling that attaches to the shaft of joint two.

All of our control computations are performed on a Pentium PC with a D/A card and an encoder interface card. Using the standard software library routines supplied with these interface cards we are able to program control algorithms directly in C.

4 The Second Plant's Model

Figure 4 shows a drawing of the Pendubot™. Since our device is a two link robot (with only one actuator) its dynamic equations can be found in numerous robotics textbooks as:



$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + h_1 + \phi_1 = \tau \quad (5)$$

$$d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + h_2 + \phi_2 = 0 \quad (6)$$

Fig. 4. Front perspective drawing of the Pendubot.

where q_1 , q_2 are the joint angles and τ is the input torque. The important distinction then between the equation 5, equation 6 and a standard two-link robot is, of course, the absence of a control input torque to equation 6. Underactuated mechanical systems generally have equilibria which depend on both their kinematic and dynamic parameters. If the Pendubot™ is mounted so that the joint axes are perpendicular to gravity, then there will be a continuum of equilibrium configurations. These equilibria are characterized by the second link vertical for any position of the first link.

5 The Neuro-controller Based on a Linear Model Using a MLP and CMAC Architecture

The direct-inverse architecture described above is the one being used. In fig. 5 we show the graphical representation of the direct-inverse neurocontroller.

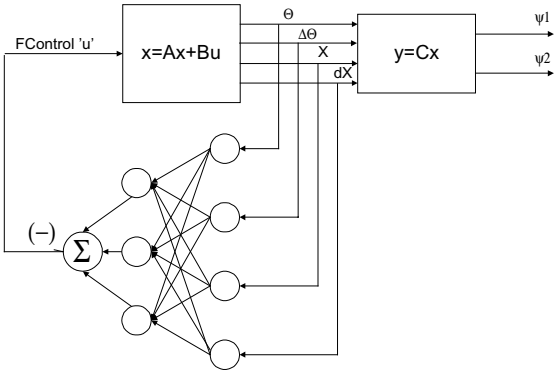


Fig. 5. Direct-Inverse architecture using an MLP.

The input/output data pairs which were generated using the linear model are used now in order to train the network. We have 4 input variables, angle, angle derivative, car position and its derivative. We have only one output, the control force acting on the inverted pendulum cart.

We have used two neural architectures to build the neurocontroller: One based in MLP network with the Levenberg-Marquart backpropagation algorithm and one based on Albus' Cerebellar Model Articulation Controller, CMAC[1]. The MLP network has three layers, with 4 units at its input layer, 3 units at the hidden layer and 1 unit at the output level. All activation functions are linear. The network training was carried out using the neural networks toolbox from MATLAB™. Training took 108 epochs for the specified error goal.

6 The CMAC Neuro-controller

The Albus CMAC network has 4 input spaces (one for each input) and one output. We have worked with a generalization parameter of 4, and a quantization vector of {20,20,1024,1024}. Our learning rate used to train the network was 0.25. The network needed 32 epochs to reach convergence. The system required only 107 memory locations because a hashing memory function is in use thus avoiding memory collisions.

The operation of the Albus CMAC can most easily be described in terms of a large set of overlapping, multi-dimensional receptive fields with finite boundaries [1]. The response of the CMAC neural network to a given input is the average of the responses of the receptive fields excited by that input, and is not affected by the other receptive fields.

Similarly, neural network training for a given input vector affects the adjustable parameters of the excited receptive fields, but does not affect the parameters of the remaining majority of receptive fields. The organization of the receptive fields of a typical Albus CMAC neural network with a two-dimensional input space is next described. The total collection of receptive fields is divided into C subsets or layers,

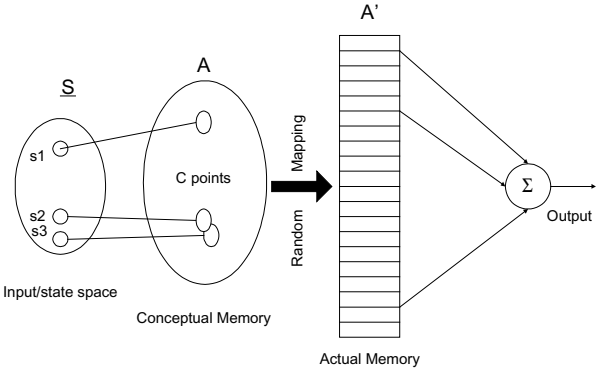


Fig. 6. Set oriented block diagram of CMAC.

which represent parallel N-dimensional hyperspaces for a network with N inputs. The receptive fields in each of the layers have rectangular boundaries and are organized so as to span the input space without overlap. Any input vector excites one receptive field from each layer, for a total of C excited receptive fields for any input.

Each of the layers of receptive fields is identical in organization, but each layer is offset relative to the others in the input hyperspace. The width of the receptive fields produces input generalization, while the offset of the adjacent layers of receptive fields produces input quantization [8]. The ratio of the width of each receptive field (input generalization) to the offset between adjacent layers of receptive fields (input quantization) must be equal to C for all dimensions of the input space. The integer parameter C is referred to as the generalization parameter. This organization of the receptive fields guarantees that only a fixed number, C, of receptive fields is excited by any input.

However, the total number of receptive fields required to span the input space can still be large for many practical problems. On the other hand, it is unlikely that the entire input state space of a large system would be visited in solving a specific problem. Thus it is not necessary to store unique information for each receptive field.

Following this logic, most implementations of the Albus CMAC include some form of pseudo-random hashing, so that only information about receptive fields that have been excited during previous training is actually stored. Each receptive field in the Albus CMAC is assumed to be an on-off type of entity. If a receptive field is excited, its response is equal to the magnitude of a single adjustable weight specific to that receptive field. If a receptive field is not excited, its response is zero.

The CMAC output is thus the average of the adjustable weights of the excited receptive fields. The Albus CMAC neural network thus produces piece-wise constant outputs. Network training is typically based on observed training data pairs $\underline{S}$ and yd($\underline{S}$), where yd(S) is the desired network output in response to the vector input $\underline{S}$. The value W (the weights) is added to each of the C memory locations W[A'j] accessed in the computation of y($\underline{S}$). This is equivalent to the well known LMS adaptation rule for linear adaptive elements with a constant training gain.

7 Application Results

We can compare two interesting features of the three architectures. First, the control signal generated by the neurocontroller and applied to the simple inverted pendulum model. The experiment consists of offsetting the pendulum by 0.1 radians and letting each of the controllers bring the pendulum to the vertical (stable) position. Figure 7 shows the control signal from the three controllers given in terms of force. The continuous line is the CMAC controller, the 'o' sign is the MLP controller and the '+' is the linear controller.

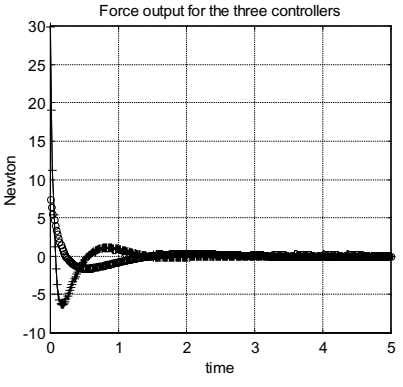


Fig. 7. Force control signal from the three controllers.

The MLP neurocontroller has a smoother response and less overshoot, while the CMAC controller has learnt the function almost exactly. Note that the CMAC force signal is oscillating over the set point which is not desired. Figure 8, shows angle evolution from 0.1 radians in the three cases. This means the inverted pendulum is started from 0.1 radians off the vertical stable line and the controller streightens the pendulum. The three controllers show different overshoot and they all reach set point at different times (settling time is different), with the CMAC neurocontroller being the best in this respect. Again in figure 8, the continuous line is the CMAC controller, the 'o' sign is the MLP controller and the '+' is the linear controller.

The problem of swinging the Pendubot up from the downward configuration to the inverted configuration is an interesting and challenging nonlinear control problem. We have used two CMAC neurocontrollers to swing up the Pendubot, i.e., to move from the stable equilibrium $q1 = -\pi/2$, $q2 = 0$, to the inverted position $q1 = \pi/2$, $q2 = 0$. The same approach can easily be modified to swing the system to any point on its equilibrium manifold. The control is switched to a second control to balance the Pendubot about the equilibrium whenever the swing up controller moves it into the basin of attraction of the balancing controller. The experimental results have been very good. In the swing up neurocontroller we used a CMAC network with 80 memory locations, 32 as generalization parameter and 0.6 as learning rate. In the balancing controller we used a CMAC network with 80 memory locations, 64 as generalization

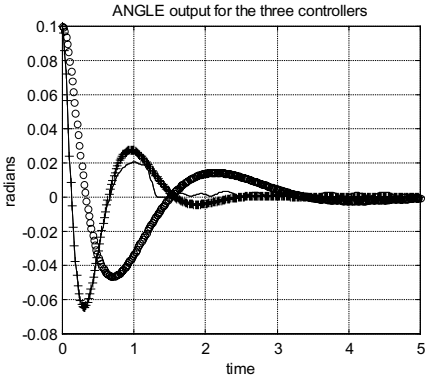


Fig. 8. Angle output from ther three controllers.

parameter and 0.7 as learning rate. Figure 9 shows the CMAC torque response after training and controlling the swing up problem in the Pendubot. There are 115 sampled points. Note that points from 1 to 40 are points from the swing up controller, while points 40 to 115 are from the balancing controller. The continuos line represents the linear controller while dashed line represents the CMAC neurocontroller´s response.

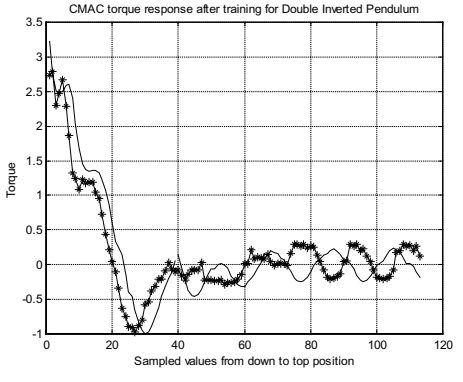


Fig. 9. Torque response of the linear and CMAC controllers.

8 Conclusions

This work shows results from three neurocontrol systems for the single pendulum and a neurocontrol scheme for a 2-DOF manipulator. We project to reach the development of a hybrid neurocontroller scheme using Pendubot. We have developed the double inverted pendulum´s model and we are in the process of developing an hybrid neurocontroller scheme based on the results above. This work has shown the feasibility of the design and construction of a neurocontroller based in a direct inverse scheme,

and it will allow us to continue developing towards the main objective. Code implementations and simulation using these models and control schemes, have introduced us to the capabilities and limitations of the CMAC algorithm. This work has also shown the feasibility of a neurocontroller based on CMAC.

References

[1] Albus, J.S., "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)", *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, v.97 (September):220-227, 1975.

[2] Åström K.J., Wittermark B., "Adaptive Control", Addison Wesley, Reading, MA., 1989.

[3] Barto A.G., "Connectionist Learning for Control", in Neural Networks for Control, Bradford Books, 1990.

[4] Brown and Harris, C.J., "Neurofuzzy Adaptive Modelling and Control", Prentice Hall, Series in Systems and Control Engineering, 1994.

[5] Hunt K.J., Sbarbaro-Hofer D., Zbikowski R.,Gawthrop P.J., "Neural Networks for Control Systems- A survey", *Automatica*, Vol. 28, pp. 1083-1112, 1992.

[6] Tolle H., Ersü E., "Neurocontrol: Learning Control Systems, Inspired by Neuronal Architectures and Human Problem Solving", *Lecture Notes in Control and Information Sciences*, 172, Springer-Verlag, Berlin, 1992.

[7] Widrow B., Stearns S.D., "Adaptive Signal Processing", Prentice Hall, Englewood Cliffs, NJ, 1985.

[8] W.T. Miller; F.H. Glanz and L.G. Kraft; “Application of a general learning algorithm to the control of robotic manipulators.” *International Journal of Robotic Research*, Vol. 6, pp. 84-98, 1987.

[9] Spong, M.W., Block D.J., Pendubot Instalation Manual and User Guide, *Mechatronic Systems, Inc*. 1997.

A New Approach for the Solution of Multiple Objective Optimization Problems Based on Reinforcement Learning

Carlos Mariano[1] and Eduardo Morales[2]

[1] Instituto Mexicano de Tecnología del Agua,
Paseo Cuauhnáhuac 8532, Jiutepec, Morelos, 62550, Mexico.
cmariano@tlaloc.imta.mx
[2] ITESM – Campus Morelos, Paseo de la Reforma 182-A,
Temixco, Morelos, 62589, Mexico,
emorales@campus.mor.itesm.mx

Abstract. Many problems can be characterized by several competing objectives. Multiple objective optimization problems have recently received considerable attention specially by the evolutionary algorithms community. Their proposals, however, require an adequate codification of the problem into strings, which is not always easy to do. This paper introduces a new algorithm, called MDQL, for multiple objective optimization problems which does not suffer from previous limitations. MDQL is based on a new distributed Q-learning algorithm, called DQL, which is also introduced in this paper. Furthermore, an extension for applying reinforcement learning to continuos functions is also given. Successful results of MDQL on a continuos non restricted problem whose Pareto front is convex and on a continuos non-convex problem with restrictions are described.

1 Introduction

Many problems can be characterized by several non-commensurable and often competing measures of performance or objectives. The multiobjective optimization problem is, without loss of generality, the problem of simultaneously minimizing the n components $f_k, k = 1, \ldots, n$, of a vector function $\mathbf{f}$ of a variable $\mathbf{x}$ in a universe u, where:

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_n(\mathbf{x}))$$

The problem has usually no unique, perfect solution, but a set of equally efficient, or non-inferior, alternative solutions, known as the Pareto-optimal set. Still assuming a minimization problem, inferiority is defined as follows:
(inferiority) *A vector* $\mathbf{u} = (u_1, \ldots, u_n)$ *is said to be inferior to* $\mathbf{v} = (v_1, \ldots, v_n)$ *iff* $\mathbf{v}$ *is partially less than* $\mathbf{u}$ $(\mathbf{v}_P < \mathbf{u})$, *i.e.,*

$$\forall i \in \{1, \ldots, n\}, v_i \leq u_i \wedge \exists i \in \{1, \ldots, n\} \mid v_i < u_i$$

Alternatively, $\mathbf{v}$ *can be said to be superior to, or to dominate,* $\mathbf{u}$.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 212–223, 2000.

(non-inferiority) *Vectors* $\mathbf{u} = (u_1, \ldots, u_n)$ *and* $\mathbf{v} = (v_1, \ldots, v_n)$ *are said to be non-inferior to each other if neither* $\mathbf{v}$ *is inferior to* $\mathbf{u}$ *nor* $\mathbf{u}$ *is inferior to* $\mathbf{v}$.

There is a especial interest in finding or approximating the Pareto-optimal set, mainly to gain deeper insight into the problem and knowledge about alternate solutions. The evolutionary algorithms (EAs) community has been specially interested in the generation of Pareto fronts as they evolve populations of candidates solutions, rather than a single point in the search space [3,11]. Despite its apparent success, EAs require a codification of the problem which is not always easy to obtain, sacrificing, in many cases, an accurate representation of the problem. Additionally the parameters associated with the EA approach, such as: weight factors used in function aggregation, priorities used in lexicographic ordering and Pareto ranking, niche size in niche induction techniques, tournament size, etc. (e.g., see [3]), need to be established. Finding the appropriate values for the parameters in EAs is most of the times as difficult as the main problem solution.

Reinforcement learning (RL) has recently received considerable attention. In particular, in RL an autonomous agent learns an optimal policy that outputs an appropriate action given the current state [10] and can be easily formulated to solve optimization problems. The first proposed algorithm, to our knowledge, to solve multiple objective optimization problems based on RL was MOAQ [7,8]. MOAQ can be seen as a natural extension of Ant-Q [5] for solving multiple objective optimization problems. It suffers, however, from similar restrictions as Ant-Q, namely, it assumes that a domain dependent heuristic is available and that a suitable combination with the evaluation function can be determined. It also requires the definition of extra paramenters which need to be tuned for each particular problem and looses all the theoretical results from reinforcement learning. Despite all these, MOAQ was used to successfully solve a highly restricted and complex multiple objective optimization problem [7] and other problems reported in the EAs community [8].

This paper introduces a new algorithm, called MDQL, which eliminates the limitations encountered by MOAQ and other Ant-Q based algorithms. MDQL is based exclusively on Q-learning and eliminates all the extra parameters introduced in Ant-Q. At the heart of MDQL is DQL, a distributed version of Q-learning also introduced in this paper. A third contribution of this paper is the adaptation of the work of Deniz Yuret [14] to handle real-value functions in the context of RL. MDQL was tried on two test cases suggested in the literature with very promising results.

Section 2 provides a brief description reinforcement learning and Q-learning, describes DQL, MDQL, and their extensions for handling real value functions. Section 3 describes the problems and experimental results where MDQL was tested. Finally, conclusions and possible future work are given in section 4.

2 Reinforcement Learning

In reinforcement learning, an autonomous agent learns an optimal policy $\pi :\rightarrow A$, that outputs an appropriate action $a \in A$, given the current state $s \in S$, where A is the set of all possible actions in a state and S is the set of states. The available information to the agent is the sequence of immediate rewards $r(s_i, a_i)$ for all the possible actions and states $i = 0, 1, 2, \ldots$. Given this kind of training information it is possible to learn a numerical evaluation function defined over states and actions, and then implement the optimal policy in terms of this evaluation function. The value of the evaluation function $Q(s, a)$ is the expected reward when starting in state s, taking action a, and thereafter following policy π.

One of the most important breakthroughs in reinforcement learning was the development of the temporal difference algorithm known as *Q-learning* [13]. Its simplest form, one step Q-learning, is defined by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{1}$$

where, $Q(s, a)$ is the action value function of taking action a in state s, $Q(s', a')$ is the action value function in the resulting state (s') after taking action a, α is the step-size parameter, r is the reward, and γ is a discount rate. The action value function is updated using information from the current value, the associated reward, and the best next action value function.

In this case the learned action-value function, Q, directly approximates Q^*, the optimal action-value function, independently of the policy being followed [10]. Q-learning requires, a *behavior* policy which determines which state-action pairs will be selected and updated, and an *estimation* policy used to update the evaluation functions for state-action pairs. The behavior policy can be stated as a *greedy*, *ε-greedy*, *Softmax,* or any other action selection policy [10], while the estimation policy is generally *greedy*. This ensures that all pairs continue to be updated obtaining correct convergence to the optimal action-value function Q^*. The Q-learning algorithm is shown in Table 1, where the number of episodes corresponds to the number of trials.

Table 1. Q-learning: algorithm

```
Initialize Q(s, a) arbitrarily
Repeat (for n episodes)
    Initialize s
    Repeat (for each step of episode)
        Choose a from s using policy derived from Q (e.g., ε−greedy)
        Take action a, observe r, s'
        Q(s, a) ← Q(s, a) + α [r + γ max_a' Q(s', a') − Q(s, a)]
        s ← s';
    until s is a terminal state
```

Table 2. Distributed Q-learning: algorithm

```
Initialize Q(s, a) arbitrarily
Repeat (for n episodes)
    Repeat (for m agents)
        Initialize s
        Repeat (for each step of episode)
            Choose a from s using policy derived from Q (e.g., ε−greedy)
            Take action a, observe r, s'
            Q(s, a) ← Q(s, a) + α [r + γ max_a' Q(s', a') − Q(s, a)]
            s ← s';
        until s is terminal
    Evaluate the m proposed solutions
    Assign rewards to the best solution found
```

In RL, the associated rewards are generally fixed in advance and are used to guide the agent towards the optimal policy. If we want to use RL for optimization problems, however, the associated rewards cannot be pre-determined as we do not know where is the optimum state. An alternative approach, is to start with some associated rewards, use an agent to try to find a solution, and adjust the rewards of the states associated with that solution accordingly to how good the solution is compared with previous solutions. This is the strategy followed by the DQL and MDQL algorithms described in the following sections.

2.1 Distributed Reinforcement Learning

Instead of a single agent, let us consider a set or family of agents trying to find an optimal policy in a common environment. In this setting, all the agents are exploring different options, all have the same goals, and all use the same policy, which is updated by all agents. This distributed algorithm, which we called DQL, can be seen as performing Q-learning m times, and then adjusting the rewards or punishments after all the m agents have reach a solution or completed a pre-determined number of steps (episode). DQL is described in Table 2. This algorithm has shown good convergence rates and can be easily parallelized.

2.2 DQL for Solving Multiple Objective Optimization Problems

The Q-learning algorithm can be further extended to multiple objective optimization problems. The idea is to assign a family of agents to each objective. The solutions obtained by each family are compared and compromise solutions are established to benefit all the objectives. This mechanism was originally proposed and tested in [7] and [8] using Ant-Q [5]. The limitations and extra parameters associated with Ant-Q based algorithms lead us to propose a more sounded algorithm based exclusively on Q-learning. This algorithm, called MDQL, is described in Table 3.

Table 3. Distributed Q-learning for multiobjective optimization problems.

```
Initialize Q(s, a) arbitrarily
Repeat (for n episodes)
    Initialize s
    Repeat (for m agents)
        Repeat (for f families)
            Repeat (for each step of episode)
                Choose a from s using policy derived from Q (e.g., ε−greedy)
                Take action a, observe r, s'
                Q(s, a) ← Q(s, a) + α [r + γ max_a' Q(s', a') − Q(s, a)]
                s ← s';
            until s is terminal
    compromise_solution=negotiation(f solutions)
    if (compromise_solution is non-dominated)
        Pareto = Pareto + compromise_solution
        Apply reward to compromise_solution
```

Table 4. Negotiation mechanism.

```
negotiation(m solutions)
Let compromise_solution=NULL
Repeat (for each solution i)
    Repeat (for each objective j)
        if (objective_j(solution_i) achieves all objective_j(solution_j))
        compromise_solution=best_of(solution_i,compromise_solution)
if (compromise_solution=NULL)
    compromise_solution=random(m solutions)
return compromise_solution
```

Each agent of each family proposes a solution for the objective that it is optimizing. The proposed solution of the i-th. agent of all the families (i.e., a solution for all the objectives) is evaluated and compared against the rest of the $m-1$ solutions proposed by the rest of the agents. A negotiation mechanism, described in Table 4, is responsible for the creation of Pareto sets. The associated rewards of the states involved in the solutions are updated according to their evaluation function (this is described in the following section). By delaying the adjustment in the rewards MDQL achieves a better distribution of solutions (Pareto set) while keeping its exploration capabilities.

In the negotiation mechanism all solutions are evaluated for all the objectives. If a solution satisfies all the objectives it is considered as a candidate compromise solution. If there is more than one possible solution, the best one is returned. Ties are broken arbitrarily. *Compromise_solution* follows a non-dominant criterion to construct a Pareto front. Non-dominated solutions are kept in the Pareto front and rewards are assigned to their components. In case where there in no

solution which dominates the rest, a new solution is chosen randomly from the rest of the solutions at that episode. The state-action evaluation functions are incrementally updated, that is, each time an agent of one family travels from one state to another. The number of episodes and steps are given arbitrarily, as long as a sufficiently large number is chosen to ensure convergence.

2.3 Application to Continuos Functions

Reinforcement learning has been generally applied to discrete problems. George Bilchev in [1,2] proposed an adaptation of the Ant Colony model for continuos problems. His proposal considers for each state a set of possible directions for the trayectory of agents. The magnitude of this trayectory is bounded by an upper limit. The selection of the trayectories and magnitudes is performed randomly at the beginning and then the most frequently selected directions are assigned higher probabilities of selection.

In an earlier work, Deniz Yuret [14] proposed an algorithm, named Dynamic Hill Climbing (DHC), which uses a similar mechanism. DHC is based on three main heuristics: storage of local minimum and restarts from distant points from those points, adjustment of the magnitudes of the trayectories used during the search, and use of successful trayectories for faster climbings (descendings).

In this paper, we take ideas from both proposals to deal with continuos functions. Each state has a set of possible actions which depends on the problem to be solved. The actions are associated with the directions that an agent can take, which depends on the dimensionality of the problem to be solved. For one dimension there are two possible directions, for 2 there are 8 (considering both directions for each dimension and all the directions of the combinations of them), for 3 there are 26, and so on (see Figure 1).

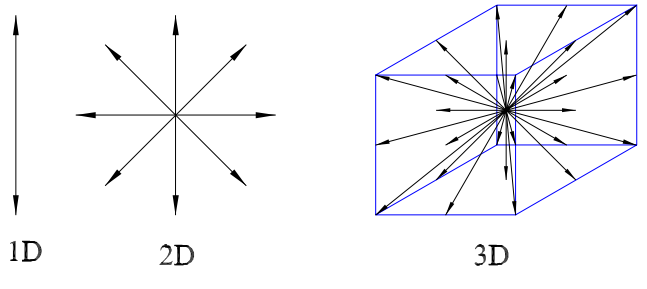


Fig. 1. The number of combinations of main directions increases exponentially with the number of dimensions.

The magnitudes of the trayectories are adjusted according to their evaluation. If a state-action pair increases the evaluation function, its magnitude is doubled, otherwise it is divided by 2. The initial magnitude is given arbitrarily. The magnitude of the trayectories represent the rewards that the agent receives when selecting a state-action pair (r in equation 1). This adjustment is necessary for solving optimization problems with RL since the optimal state is not known in advance.

3 Experiments

This paper shows the results of MDQL on two multiple objective optimization problems suggested in [12].

3.1 Shaffer's F2

The first case is a non-restricted problem originally proposed by Shaffer [9], that has beed used to test many of the algorithms proposed for solving multiple objective problems. It has a convex Pareto front and only one decision variable:

$$\begin{aligned} &\text{Minimize } F = (f_{21}(x), f_{22}(x)), \text{ where} \\ &\quad f_{21}(x) = x^2, \\ &\quad f_{22}(x) = (x-2)^2, \end{aligned}$$

where $-6.00 \leq x \leq 6.00$.

Figure 2 shows f_{21} and f_{22}. It is clear from the figure that the Pareto front is where the tradeoff between the two functions is. That is, for $0 \leq x \leq 2.00$.

The reported solutions to this problem using genetic algorithms have used niches [3] or non generational behavior [11] to maintain distributed solutions over the Pareto front. For this problem, 240 states in the range of x were considered. These states represent a discretization over the continuos space over which the rewards will be adjusted incrementally. There are only two possible actions (since there is only one decision variable). Each state-action pair has its own Q value and its reward r (in case it is selected). In both cases initial values of 1 for Q and r were considered. The experiments were run with 200 episodes, 100 steps per episode, and 100 agents per family. The initial state for both families is $x = 0$. A discount factor of $\gamma = 1$ and a learning step of $\alpha = 0.1$ were used in the simulation.

Figure 3 shows the Pareto front obtained with MDQL. Figure 4 shows the Pareto solutions in the Pareto front, which has more evenly distributed solutions when compared against our previous proposal (MOAQ) [8] and the solutions reported by [3] and [11] using genetic algorithms.

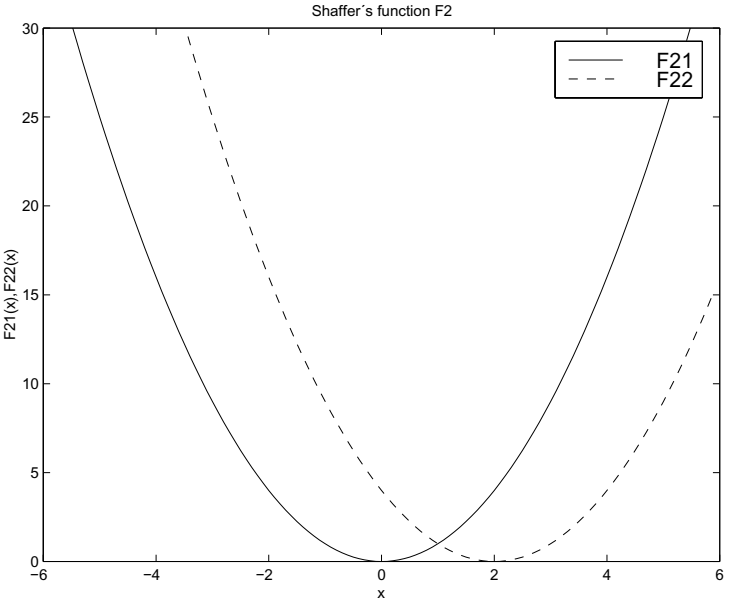


Fig. 2. Shaffer's function F2.

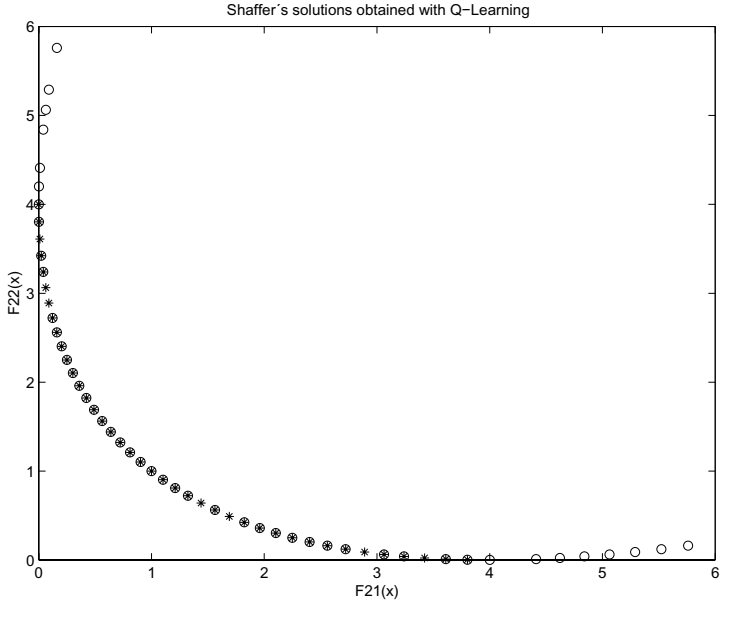


Fig. 3. True vs. calculated Pareto front for Shaffer F2 obtained with MDQL, * analytical, ∘ calculated Pareto front.

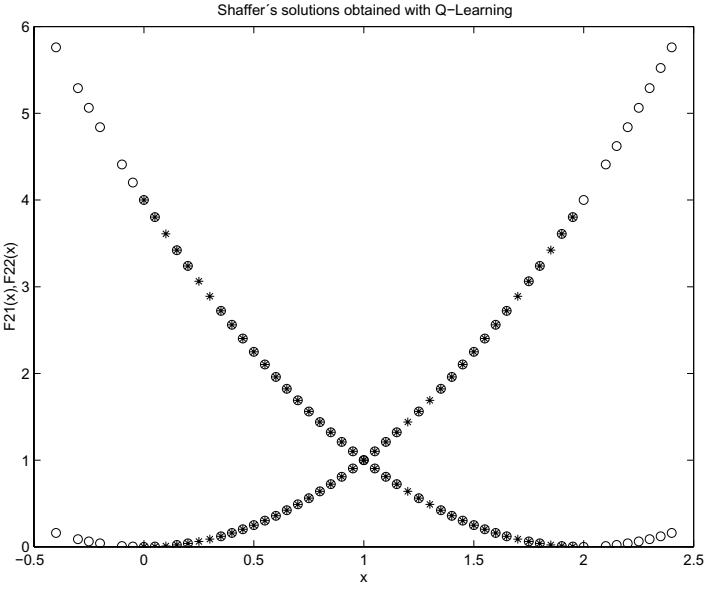


Fig. 4. Pareto solutions obtained with MDQL.

3.2 Fonseca's Problem 2

The second test case was originally proposed by Fonseca [4]. It considers two objective functions and can include additional decision variables without changing the Pareto front (scability). It has a non-convex Pareto front:

$$\text{Minimize } F = (f_1(\vec{x}), f_2(\vec{x})), \text{ where}$$

$$f_1(\vec{x}) = 1 - \exp\left(-\sum_{i=1}^{n=2}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right),$$

$$f_2(\vec{x}) = 1 - \exp\left(-\sum_{i=1}^{n=2}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right),$$

where $\vec{x} = (x_1, x_2)$, and $-2 \leq x_1, x_2 \leq 2$.

This is a more difficult problem, as it has a greater search space, there are two decision variables $\vec{x} = (x_1, x_2)$, and has a non-convex Pareto front. In this case, there are 8 possible actions for each agent and 800 states were considered for each component of $\vec{x}$. As in the previous problem the values for Q and r were initially set to 1. The experiments were run with 300 episodes, 400 steps per episode, and 100 agents per family. The initial state is $x_1 = x_2 = 0$ for both families and γ and α were set to the same values ($\gamma = 1$, $\alpha = 0.1$). Figures 5 and 6 show the results for the best of ten trials. These results are not as tightly close to the Pareto-optimal set as in the previous problem, nevertheless, they show a nice distribution around it.

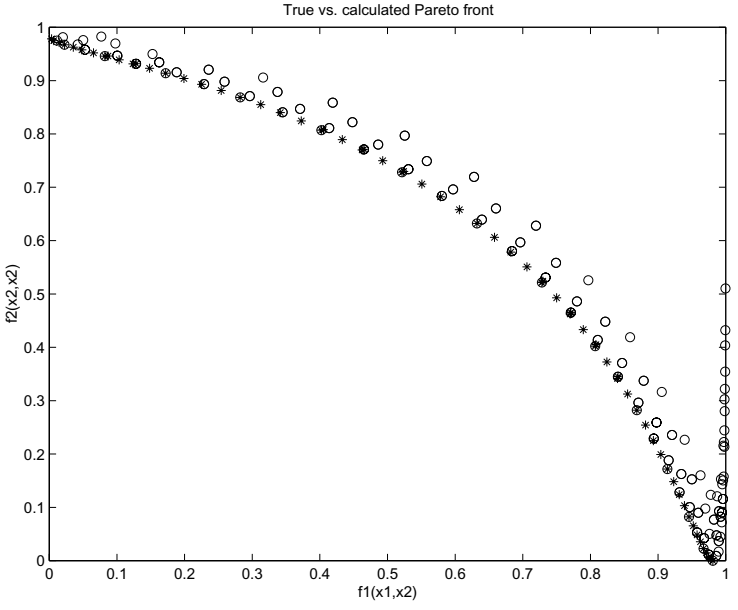


Fig. 5. True vs. calculated pareto front for Fonseca's F2 function, * analytical, ∘ calculated.

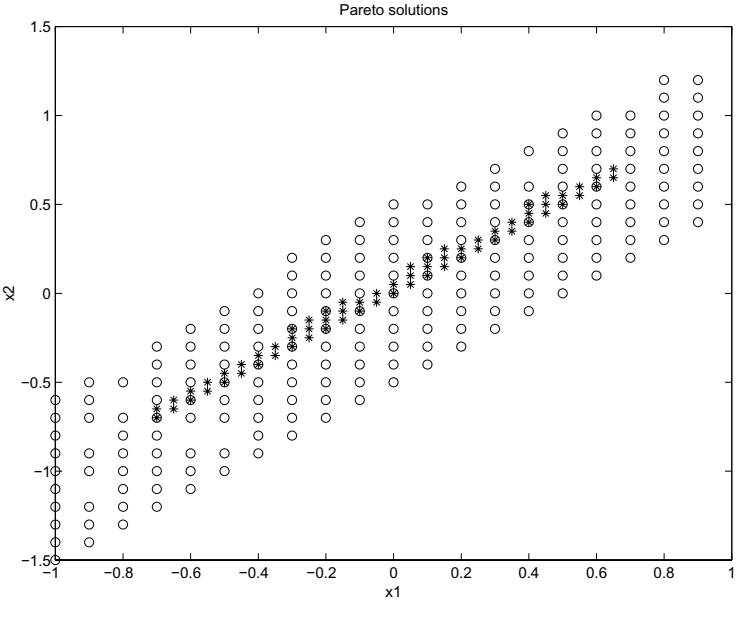


Fig. 6. Pareto solutions, * analytical solutions, ∘ solutions found.

The convergence times for both problems are around 50 seconds for Shaffer's F2 and 85 seconds for Fonseca's problem 2[1].

4 Discussion and Future Work

The solution of multiple objective optimization problems has recently received considerable attention, specially by the genetic algorihtm's community. In this paper, a new algorithm, called MDQL, based on Q-learning that can be used for multiple objective optimization problems was introduced. MDQL is based on a distributed Q-learning algorithm whose rewards are updated incrementally depending on the evaluation function of the agents. It was shown how MDQL was used on two test problems, outperforming a previous algorithm that was designed for the same tasks with extra parameters.

As can be seen in Shaffer's F2 problem convergence time was reduced considerably with respect to MOAQ, unfortunately previous works using genetic algorithms does not report convergence times, so a comparison with this parameter cannot be made. Our future work will continue testing MDQL on the rest of the test cases suggested in [12] and using metrics of comparison suggested by van Veldhizen and by other authors [15,6].

References

1. George Bilchev and Ian Parmee. The Ant Colony Methaphor for Searching Continuous Design Spaces, in T. Fogarty, ed., Lecture Notes in Computer Science 993, Springer-Verlang, 1995, pp. 25-39.
2. George Bilchev and Ian Parmee. Constrained optimization with an Ant Colony search model. In *Proceedings of ACEDC'96*, PEDC, University of Plymouth, UK-26-28 March, 1996
3. Carlos M. Fonseca and Peter J. Flemming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation,* 3(1):1-16, 1995.
4. Carlos M. Fonseca and Peter J. Flemming. Multiobjective optimization and Multiple Constraint Handling with Evolutionary Algorithms I: A Unified Formulation. *Research Report 564,* Dept. Automatic Control and Systems Eng., University of Sheffield. Sheffield S1 4Du, U.K., January, 1995.
5. Gambardella, L.M., Dorigo, M. (1995) Ant-Q: A Reinforcement Learning approach to the traveling salesman problem. In *Proceedings of the 12th. International Machine Learning Conference* (ML-95), pp., 252-260, Morgan Kaufmann.
6. Joshua Knowles and David Corne. Assesing the Performance of the Pareto Achieved Evolution Strategy. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (eds.). *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference Workshop Program* (pp. 123-124). July 13-17, 1999, Orlando, Florida USA. San Francisco, CA: Morgan Kaufmann.

[1] MDQL is coded in Borland C++ ver. 4.5 and runs of a PC Pentium MMX at 200 MHz.

7. Mariano, C., Morales E. (1999). MOAQ an Ant-Q Algorithm for Multiple Objective Optimization Problems. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (eds.). *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 894-901). July 13-17, 1999, Orlando, Florida USA. San Francisco, CA: Morgan Kaufmann.
8. Mariano, C., Morales E. (1999). MOAQ a Distributed Reinforcement Learning Algorithm for the Solution of Multiple Objectives Optimization Problems. in *Memorias del Segundo Encuentro Nacional de Computación ENC99*, paper No. 111 . 12-15 Sept, Pachuca Hidalgo, Mexico, Sociedad Mexicana de Ciencias de la Computación.
9. Schaffer, D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In J.J. Grefenstette (Ed.), *Genetic Algorithms and Their Applications: Proccedings of the First International Conference on Genetic Algorithms* (pp. 93-100). Hillsdale, NJ: Lawrence Erlbaum.
10. Sutton, R.S., Barto, A.G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.
11. Manuel Valenzuela-Rendón and Eduardo Uresti-Charre. A non-generational genetic algorithm for multiobjective optimization. In *Proceedings of the Seventh International conference on Genetic Algorithms,* pp. 658-665, San Francisco, California, 1997, Morgan Kaufmann.
12. Dave van Veldhizen and Gary Lemont. MOEA Test Suite Generation, Design & Use. *Research Report,* Department of Electrica and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Ohio, June 1999.
13. C.J.C.H. Watkins. Learning from Delayed Rewards. Ph.D. thesis, Cambridge University. 1989.
14. Deniz Yuret. From Genetic Algorithms to Efficient Optimization, *Msc Thesis in Electrical Engineering and Computer Science,* Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, May 1994.
15. Eckart Zitzler and Lothar Thiele. Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. In *Proceedings of the Parallel problem Solving from Nature- PPSN V* pp. 292-301. Amsterdam The Netherlands, September 1998, Springer.

Automatic Extraction of Logic Program Transformations from Examples

Wamberto Weber-Vasconcelos* and Marcelo A.T. Aragão

[1] Departmento de Estatística e Computação
Universidade Estadual do Ceará
Ceará, Brazil
wvasconcelos@acm.org
[2] Departmento de Informática
Banco Central do Brasil
Brasília, Brazil
marcelo.aragao@bcb.gov.br

Abstract. The design of program transformations is a difficult and error-prone task. We propose a practical approach to help programmers devise their program transformations: the programmer is to supply two versions of the same logic program, an inefficient one and another with a better computational performance, and the set of syntactic adjustments required to map them is automatically found. These adjustments are represented as transformations using an extended language based in program schemata and are subsequently made more generic, becoming applicable to a larger class of programs and thus extending the benefits of the transformation.

1 Introduction

Research on program transformation has proposed and validated methods for developing correct program transformations (e.g., [6], [8] and [2]). The *correctness* of a transformation refers to whether the meaning of a program (expressed in some form of semantics) is preserved after the alterations prescribed by the transformation are applied to it [7].

Useful program transformations, on the other hand, are those of generic applicability, i.e. they are not too specific and hence a larger class of programs can benefit from them, and of practical importance, i.e. they contemplate inefficient program constructs actually (and frequently) found. The process of devising such useful program transformations consequently involves much ingenuity and creativity. Furthermore, program transformations must be represented in some sort of meta-level formalism since classes of programs in an object language are addressed. All these factors make the design of program transformations a difficult and error-prone task.

* Partially sponsored by the Brazilian National Research Council (CNPq) grant no. 352850/96-5.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 224–234, 2000.

To circumvent these difficulties, we propose a practical approach to help programmers devise their program transformations. Rather than manipulating an abstract formalism to represent transformations, we advocate that an example, comprising two versions of the same program (an inefficient one and another with a better computational performance), be supplied and the set of syntactic adjustments necessary to map these two versions be automatically obtained. These syntactic adjustments are then made more *generic* by removing irrelevant details such as particular names of predicates and functors, sequences of subgoals that do not change, and so on, thus becoming applicable to a larger class of programs similar to the initial examples. The benefits of the transformation extracted from a single example (comprising a pair of programs) is extended to a number of distinct programs.

In order to represent these syntactic adjustments, we have used a schema-based language, a variant of the formalism proposed in [3] and enhanced in [13], and extended it with *constraints*, as developed in [2]. This formalism allows the declarative representation of a large repertoire of programs and syntactic modifications. In Section 2 below we show in more detail the formalism employed.

In Section 3 we describe how, given a pair of programs (the original one and its improved version), our system extracts a transformation. We characterise the class of program transformations that can be successfully extracted by our system. In the same section we describe the generalisation procedure. In Section 3.2 we show a working example of a transformation being extracted from a pair of programs and its generalisation. In Section 4 we draw some conclusions and discuss related issues.

2 Representing and Using Transformations for Logic Programs

The efficiency issues of a program can be dealt with either during its design or at a later stage, when the finished program is analysed and its inefficient constructs are detected and appropriately modified, conferring a better computational performance on it. Our work follows the latter approach and we advocate the use of *program transformations* for that purpose. Program transformations consist of the description of inefficient program constructs and how they should be altered (so as to confer a better computational behaviour on the program) using a convenient notation. Such transformations can be used to exhaustively analyse a program and find opportunities for improving it [13]. In this work we propose an automatic approach to obtain the program transformations which are going to be used in such a context.

2.1 Schema-Based Language with Constraints

In order to represent program transformations, we shall make use of the schema-based language defined in [13] after the original proposal in [3]. Such a formalism allows us to highlight important features of a program and to disregard its

irrelevant details. However, we shall also incorporate extra *constraints* into our language: these were originally suggested in [13] and developed in [2].

Program schemata are sequences of schematic Horn clauses. A schematic clause is a clause with schematic literals in it. A schematic literal is i) a literal whose predicate symbol is a second-order variable with a sequence of schematic terms; or ii) a (conventional) literal with a sequence of schematic terms; or iii) the construct $\boldsymbol{G}_i$ which stands for a sequence i of terms; the positive literal (i.e., the head goal) is not allowed to be a $\boldsymbol{G}_i$. A schematic term is i) a term whose function symbol is a second-order variable with a sequence of schematic terms; or ii) a (conventional) function symbol with a sequence of schematic terms; or iii) the construct $\boldsymbol{A}_i$, which stands for a sequence of literals; or iv) any conventional (first-order) term. It should be noticed that any conventional logic program is a schema, but a very specific one which only allows for different instances as far as variable renaming is concerned.

We associate with each program schema a finite set of *constraints* on its constructs. This proposal has been profitably pursued in [2] (after a suggestion in [13]): in this way classes of programs more precisely defined can be obtained. Restrictions, other than syntactic ones, on the schemata can be conveniently expressed. We shall employ the repertoire of *global constraints* of [2] – these can be *form constraints* – predicates $constant(t), ground(t), var(t)$, where the predicates have the usual Prolog meaning, and $in(l, \boldsymbol{G}), in(t, \boldsymbol{A})$, which holds if literal l or term t (schematic or not) is found in $\boldsymbol{G}$ or $\boldsymbol{A}$, respectively; or *length constraints* – predicate *length*, applicable to vectors of terms, $length(\boldsymbol{A}_i, t)$, and vectors of literals, $length(\boldsymbol{G}_i, t)$, where t is either a variable or a natural number[1]; or *combinators* – the form and length constraints can be combined forming global first-order formulae (constraints) with $\wedge$ (logical "and"), $\vee$ (logical "or") and $\neg$ (logical "not"). We have left out the *local constraints* listed in [2] since they were not necessary for our purposes here. We shall refer generically to program schemata as $\mathbf{S}$ and to global constraints as γ.

As an example, we show the following schema: it generically represents the class of all programs single-recursively manipulating data structures (such as lists or trees) with exactly two clauses:

$$\left\langle \begin{array}{l} P(\boldsymbol{A}_1, X, \boldsymbol{A}_2) \leftarrow \boldsymbol{G}_1 \\ P(\boldsymbol{A}_3, F(\boldsymbol{A}_4), \boldsymbol{A}_5) \leftarrow \boldsymbol{G}_2, P(\boldsymbol{A}_6, Y, \boldsymbol{A}_7), \boldsymbol{G}_3 \end{array} \right\rangle$$
$$\boxed{\begin{array}{l} constant(X) \wedge length(\boldsymbol{A}_1, L) \wedge length(\boldsymbol{A}_3, L) \wedge length(\boldsymbol{A}_6, L) \wedge \\ in(Y, \boldsymbol{A}_4) \wedge \neg in(P(\boldsymbol{A}), \boldsymbol{G}_1) \wedge \neg in(P(\boldsymbol{A}), \boldsymbol{G}_2) \wedge \neg in(P(\boldsymbol{A}), \boldsymbol{G}_3) \end{array}}$$

Only those programs consisting of two clauses, are represented above. Moreover, the first clause must be non-recursive (global constraint $\neg in(P(\boldsymbol{A}), \boldsymbol{G}_1)$) and the second clause must be singly recursive (constraints $\neg in(P(\boldsymbol{A}), \boldsymbol{G}_2) \wedge \neg in(P(\boldsymbol{A}), \boldsymbol{G}_3)$). The lengths of some sequences of terms are constrained to be the same ($length(\boldsymbol{A}_1, L) \wedge length(\boldsymbol{A}_3, L) \wedge length(\boldsymbol{A}_6, L)$) thus ensuring that the terms focused upon (the recursive data structure) in both head goals and in the

[1] The *length* constraint has been independently proposed in [8] and [2] and it provides an elegant alternative to the explicit reference to the position of a term proposed in [13].

recursive call occupy exactly the same position. The presence of Y in the appropriate position in the head goal of the second clause (constraint $in(Y, \boldsymbol{A}_4)$) and in the recursive call as well as the constant ($constant(X)$) in the non-recursive clause characterise the intended class of programs.

A *substitution pair* is $\diamond/\square$ where $\diamond$ is a schematic construct and $\square$ is a suitable logic programming construct, that is, $\boldsymbol{G}/\langle G_1, \ldots, G_n\rangle$, or $\boldsymbol{A}/\langle t_1, \ldots, t_n\rangle$, or P/p_i^j (a predicate variable), or F/f_i^j (a function variable), or X/t_i (a variable). A *substitution* σ is a finite set of substitution pairs $\diamond_i/\square_i$ such that no two $\diamond_i, \diamond_j, i \neq j$, are the same. We can associate to a schema a (possibly infinite) number of substitutions. The *application* of a substitution $\sigma = \{\diamond_1/\square_1, \ldots, \diamond_n/\square_n\}$, to a schema $\mathbf{S}$, denoted by $\mathbf{S}\sigma$ is obtained by the simultaneous replacement of all occurrences of $\diamond_i$ by $\square_i$ in $\mathbf{S}$.

2.2 Schema-Based Transformations

Program transformations can be conveniently represented using the schemata above. This idea was originally pursued in [3], in the context of logic languages, and earlier in [5], for functional languages. Schemata can be used to describe typical, inefficient programming constructs and how these should be changed in order to confer a better computational behaviour on those programs in which they are found.

Using the definitions above, we can formalise a particular class of program transformations as the tuple $\mathcal{T} = \langle \mathbf{G}, \mathbf{S}, \mathbf{G}', \mathbf{S}', \gamma\rangle$ where $\mathbf{G}$ and $\mathbf{G}'$ are schematic subgoals, $\mathbf{S}$ and $\mathbf{S}'$ are program schemata and γ is the set of global constraints on $\mathbf{G}$, $\mathbf{G}'$, $\mathbf{S}$ and $\mathbf{S}'$. Such transformations can be understood as "any program which matches the input schema $\mathbf{S}$ and is called as $\mathbf{G}$ should be rewritten as the output schema $\mathbf{S}'$ and its call(s) replaced by $\mathbf{G}'$, all subject to the set of constraints γ." It describes an *opportunity* to improve the computational performance of logic programs – those which match the transformation can be modified.

3 Automatic Extraction of Transformations from Examples

Our approach aims at the following scenario: an expert programmer finds an inefficient program and, after analysing it, proposes an improved version, with a better computational performance. The programmer then submits both programs and their respective calls to our tool which is capable of extracting the set of syntactic adjustments necessary to map them — this set is represented as an initial schema-based transformation specific to that pair of programs. This initial transformation undergoes a process which makes it more generic and hence applicable to a larger class of programs. The more generic transformation characterises a new opportunity to be incorporated into a (semi-) automatic program analysis and transformation tool. In this way other programs with efficiency problems are candidates for being improved with the application of this transformation.

Our framework handles a specific class of program transformations, viz., those that fit the definition above. We specifically contemplate those transformations aimed at a *single* procedure[2]. The definition for transformations given above can be thus seen as a *template* for the class of program transformations aimed at. The extracted transformations are used within a tool for the automatic analysis and transformation of logic programs. This tool adopts an opportunistic policy for analysing and transforming programs, exhaustively testing the applicability of each transformation of a given repertoire to a logic program. If a transformation is applicable then the prescribed alterations are performed on the program and the result is re-submitted for further analysis. The search for opportunities for applying the transformations can be carried out in an exhaustive fashion, as proposed in [13], or it may follow more efficient strategies, as proposed in [12].

3.1 Generalisation of Program Transformations

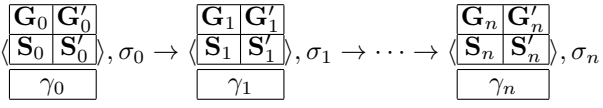
Given two sample programs, the first step is to try and match them against the definition for transformations, which is, as pointed out above, a kind of template. This process always terminates either when the template is successfully filled by the examples or when it is not possible to do so. After the matching process, there follows an independent stage, the *generalisation procedure*.

During the generalisation procedure specific details not relevant to the transformation are replaced by more generic constructs. Pairs of corresponding clauses in the schemata are compared in order to detect what was added, modified, eliminated or reordered – what remains unchanged is irrelevant to characterise the transformation, and hence should be replaced by an adequate construct of the schema language. The generalisation process attempts to map identical subgoals and replace them for vectors of literals; system tests, arithmetic expressions, constants and functors have their predicate symbols abstracted and specific user-defined names are substituted for predicate variables. A similar process takes place with literals: functors are replaced by meta-variables and sequences of literals are replaced by vectors of literals. Finally, if necessary, the sequences of calls are generalised in accordance with the predicates in the clauses. Unmodified clauses are replaced by vectors of clauses. This process is akin to the *most specific generalisation* algorithm for program schemata given in [4].

We have formally defined a generalisation procedure. It works by examining the constructs of an initial transformation and trying to build the set of constraints from the pairs of schemata. Initially, when a pair of procedures and their respective calls are matched to the definition of a transformation, the set of constraints is empty – the procedures and their calls are very specific constructs, that can only be used in the precise procedure the input schema stands for. Corresponding constructs are examined in the input and output schemata and schematic goals and a *substitution* σ is gradually obtained: the substitution pairs

[2] This, of course, can be extended to any number of procedures, but to keep the discussion simple we introduce our approach using the easier case.

relate actual program constructs that can be replaced by more generic schematic constructs. The following diagram graphically depicts the process:



Initially, $\gamma_0 = \sigma_0 = \emptyset$, for $\mathbf{G}_0, \mathbf{S}_0, \mathbf{G}'_0$ and $\mathbf{S}'_0$ are all concrete program constructs. Corresponding pairs of constructs of the input and output schemata are compared and, if appropriate, they are replaced by schematic counterparts – each replacement is recorded by means of an associated substitution. This process terminates when there are no more concrete constructs than can be replaced by generic schematic counterparts. A number of opportunities for generalisation of program transformation schemata can be detected and incorporated into this stage of the process. For instance, the actual names of user-defined predicates are not relevant, and they can be safely generalised as predicate variables, (as long as this is performed consistently within the schema); sequences of subgoals can be generalised as vectors of literals.

Once the final substitution is obtained, it is examined and some of its pairs are removed and some others are translated into global constraints. Those substitution pairs which are not relevant for characterising the transformation schemata are discarded and the generalisation is sanctioned; those pairs that are relevant are preserved.

We have come up with a simple notion of relevance: given a substitution pair $\diamond/\square$, if $\diamond$ appears unaltered both in $\mathbf{S}_i$ and $\mathbf{S}'_i$ then the pair is discarded (which means $\diamond$ is replaced, in the schema, by a most specific schematic counterpart $\square$, free of any constraints). If, on the other hand, $\diamond$ is modified or removed between $\mathbf{S}$ and $\mathbf{S}'_i$, then it should not be generalised – its substitution pair should be preserved as part of the set of constraints. For instance, a user-defined predicate that is eliminated or modified has its substitution pair translated into a global constraint in which the definition of that predicate is also incorporated. This is based on the assumption that as long as the definition of the predicate is that one being given in the constraints, then the schema unifies with the program.

A schemata transformation (program schemata and global constraints) and its substitution have been generalised if the substitution pairs have all been analysed and either translated as global constraints or discarded.

3.2 Working Example

Our approach aims at expert programmers who are responsible for supplying two versions of the same program which compute the same results but with different computational performances. As a working example, let us consider the pair of programs shown below, which solve the Towers of Hanoi problem [9].

$hanoi(A, B, C, D, E)$	$hanoi(A, B, C, D, E - [])$
$hanoi(0, O, D, T, []) \leftarrow$ $hanoi(N, O, D, T, Mvs) \leftarrow$ $\quad N > 0,$ $\quad PN\ is\ N - 1,$ $\quad CM = O/D,$ $\quad hanoi(PN, O, T, D, MvsB),$ $\quad hanoi(PN, T, D, O, MvsA),$ $\quad append(MvsB, [CM\|MvsA], Mvs)$	$hanoi(0, O, D, T, L - L) \leftarrow$ $hanoi(N, O, D, T, MvsB - MvsA) \leftarrow$ $\quad N > 0,$ $\quad PN\ is\ N - 1,$ $\quad CM = O/D,$ $\quad hanoi(PN, O, T, D, MvsB - [CM\|L]),$ $\quad hanoi(PN, T, D, O, L - MvsA)$
$\gamma_0 = \emptyset$	

$\langle$ $\rangle, \emptyset$

The procedures have as arguments, respectively, an instance of the problem (number of discs), the identification of the source peg, the identification of the final peg, the identification of the auxiliary peg and the expected sequence of disc moves. The version on right-hand side executes the same computations of its counterpart at the left-hand side, but uses a difference list technique [9] saving the call to *append/3*, which sequentially traverses the sublist *MvsB*. The pair of programs with respective calls fits our definition – they are shown in above as they would match it. As it is, the transformation above can only be applicable to the *hanoi/5* program: specific names of predicate, syntax and order of non-recursive subgoals and arguments not relevant for the prescribed alterations are represented as part of the transformation, rendering it non-applicable to other similar programs. We have also shown, in the diagram above, the initial global constraints and substitution (both empty).

In order to make the initial transformation more generic, specific details are gradually generalised and recorded as the substitution σ. During this process predicate and functor symbols are replaced by meta-variables; sequence of literals and terms are replaced by the adequate vectors, and so on. The outcome of the initial process for our example is:

$P(\boldsymbol{A}_1, E, \boldsymbol{A}_2)$	$P(\boldsymbol{A}_1, E - [], \boldsymbol{A}_2)$
$P(\boldsymbol{A}_3, X_1, \boldsymbol{A}_4) \leftarrow \boldsymbol{G}_1$ $P(\boldsymbol{A}_5, Mvs, \boldsymbol{A}_6) \leftarrow$ $\quad \boldsymbol{G}_2,$ $\quad P(\boldsymbol{A}_7, MvsB, \boldsymbol{A}_8),$ $\quad \boldsymbol{G}_3,$ $\quad P(\boldsymbol{A}_9, MvsA, \boldsymbol{A}_{10}),$ $\quad \boldsymbol{G}_4,$ $\quad Q(MvsB, F(CM, MvsA), Mvs)$ $\quad \boldsymbol{G}_5$	$P(\boldsymbol{A}_3, G(L, L), \boldsymbol{A}_4) \leftarrow \boldsymbol{G}_1$ $P(\boldsymbol{A}_5, G(MvsB, MvsA), \boldsymbol{A}_6) \leftarrow$ $\quad \boldsymbol{G}_2,$ $\quad P(\boldsymbol{A}_7, G(MvsB, F(CM, L)), \boldsymbol{A}_8),$ $\quad \boldsymbol{G}_3,$ $\quad P(\boldsymbol{A}_9, G(L, MvsA), \boldsymbol{A}_{10}),$ $\quad \boldsymbol{G}_4$ $\quad \boldsymbol{G}_5$
$\gamma_i = \emptyset$	

$\langle$ $\rangle,$

$P/hanoi, Q/append,$
$F/\cdot, G/-, X_1/[],$
$\boldsymbol{A}_1/\langle A, B, C, D\rangle, \boldsymbol{A}_2/\langle\,\rangle,$
$\boldsymbol{A}_3/\langle 0, O, D, T\rangle, \boldsymbol{A}_4/\langle\,\rangle,$
$\boldsymbol{A}_5/\langle N, O, D, T\rangle, \boldsymbol{A}_6/\langle\,\rangle,$
$\boldsymbol{A}_7/\langle PN, O, T, D\rangle, \boldsymbol{A}_8/\langle\,\rangle,$
$\boldsymbol{A}_9/\langle PN, T, D, O\rangle, \boldsymbol{A}_{10}/\langle\,\rangle,$
$\boldsymbol{G}_1/\langle\,\rangle,$
$\boldsymbol{G}_2/\langle N > 0, PN\ is\ N - 1,$
$CM = O/D\rangle,$
$\boldsymbol{G}_3\langle\,\rangle, \boldsymbol{G}_4\langle\,\rangle, \boldsymbol{G}_5\langle\,\rangle$

After the initial generalisation, the substitution pairs are examined and tested for their relevance. Our working example yields the final transformation below:

$P(\boldsymbol{A}_1, E, \boldsymbol{A}_2)$	$P(\boldsymbol{A}_1, E - [], \boldsymbol{A}_2)$
$P(\boldsymbol{A}_3, X_1, \boldsymbol{A}_4) \leftarrow \boldsymbol{G}_1$ $P(\boldsymbol{A}_5, Mvs, \boldsymbol{A}_6) \leftarrow$ $\quad \boldsymbol{G}_2,$ $\quad P(\boldsymbol{A}_7, MvsB, \boldsymbol{A}_8),$ $\quad \boldsymbol{G}_3,$ $\quad P(\boldsymbol{A}_9, MvsA, \boldsymbol{A}_{10}),$ $\quad \boldsymbol{G}_4,$ $\quad Q(MvsB, F(CM, MvsA), Mvs)$ $\quad \boldsymbol{G}_5$	$P(\boldsymbol{A}_3, G(L, L), \boldsymbol{A}_4) \leftarrow \boldsymbol{G}_1$ $P(\boldsymbol{A}_5, G(MvsB, MvsA), \boldsymbol{A}_6) \leftarrow$ $\quad \boldsymbol{G}_2,$ $\quad P(\boldsymbol{A}_7, G(MvsB, F(CM, L)), \boldsymbol{A}_8),$ $\quad \boldsymbol{G}_3,$ $\quad P(\boldsymbol{A}_9, G(L, MvsA), \boldsymbol{A}_{10}),$ $\quad \boldsymbol{G}_4$ $\quad \boldsymbol{G}_5$

$Q =_{\text{def}} \boxed{\begin{array}{l} Q([], X, X) \leftarrow \\ Q([X|Y], Z, [X|W]) \leftarrow Q(Y, Z, W) \end{array}} \wedge\ F = \cdot\ \wedge\ X_1 = []\ \wedge\ G = -\ \wedge$
$length(\boldsymbol{A}_1, L)\ \wedge\ length(\boldsymbol{A}_3, L)\ \wedge\ length(\boldsymbol{A}_5, L)\ \wedge\ length(\boldsymbol{A}_7, L)\ \wedge\ length(\boldsymbol{A}_9, L)$

The constructors generalise the portions of the programs preserved by the transformation. This final transformation is applicable to a much larger class of programs than the first original one. The actual position of the focused argument (the last one) has been appropriately generalised for it is not important: an altogether different ordering of arguments could have been used in another version of *hanoi/5*. The added vectors of terms and literals also provide extra flexibility when applying the transformation to variations of the original program.

The transformation above can be successfully applied to the left-hand side version of program *qsort/2* below, implementing the *quicksort* algorithm [9] to order a list. Although this other program has a rather different purpose than that of the *hanoi/5* and a distinct syntax, the transformation can be applied to it, conferring on it the benefits of removing a call to *append/3*. The programs before and after the application of the transformation are shown in the left-hand and right-hand side, respectively, below:

qsort(*A*,*B*)
$qsort([], []) \leftarrow$ $qsort([Cr\|Crs], Srt) \leftarrow$ $\quad partition(Crs, Cr, CrB, CrA),$ $\quad qsort(CrB, SCrB),$ $\quad qsort(CrA, SCrA),$ $\quad append(SCrB, [Cr\|SCrA], Srt)$

qsort(*A*,*B*-[])
$qsort([], L - L) \leftarrow$ $qsort([Cr\|Crs], SCrB - SCrA) \leftarrow$ $\quad partition(Crs, Cr, CrB, CrA),$ $\quad qsort(CrB, SCrB - [Cr\|L]),$ $\quad qsort(CrA, L - SCrA)$

4 Conclusions and Discussion

We propose in this work a practical approach for the automatic extraction of transformations for logic programs. By means of a single pair of examples showing parts of logic programs and how these can be modified to improve their computational performance, we automatically extract the set of syntactic alterations mapping the different versions. These alterations are then generalised, enabling the extracted transformations to be applicable to a larger set of programs than those originally intended.

Our approach does not fall within the scope of machine learning nor inductive programming. Our approach does not make use of *sets* of pairs of sample programs to train algorithms or to be used as induction hypotheses trying to obtain a transformation that encompassed all pairs of examples. Our approach differs radically: we find the transformation specific to a single pair of programs and then we generalise it, enabling it to be used with similar programs.

Our proposal was made possible mainly because a simple yet powerful notation, the schema language, was employed to record the syntactic adjustments in a concise fashion. The formalism also made possible the generalisation process, thus potentially extending the benefits to other programs than those in the original pair of examples. The opportunistic framework within which the extracted transformations are used offers additional motivation for the generalisation stage. Our approach is obviously not able to extract all possible logic program transformations. However, it formally characterises a specific class of useful program manipulations, stated as all those possible ways to fill in the definition. An attempt at filling the definition always terminates (with either success or failure) in polynomial time.

The extracted schemata represent transformations prescribing changes in the calls to procedures. A procedure call can be replaced by another call to a new procedure, which was obtained from the original ones. The new procedure is included in the program, in addition to the existing procedures, so as to guarantee extensions of the original program. A transformation can be read as: "given a program Π, replace all the calls $\mathbf{G}$ in it (whose definitions are $\mathbf{S}$), for calls $\mathbf{G}'$ and include in Π a new procedure $\mathbf{S}'$, yielding Π', a program equivalent to Π".

A special class of transformations is also captured with our definition above, viz. those prescribing only internal changes to a procedure. Its application, however, is made in a different way: the new procedure $\mathbf{S}'$ is not included in the program, but *replaces* the original procedure.

Our approach has been incorporated to the prototype AuXTran (**A**utomatic E**x**tractor of **Tran**sformations), integrated to the OpTiSB system for the efficient opportunistic analysis of programs proposed in [12] and improved in [14]. This implementation contemplates the participation of an expert programmer who should supply a pair of procedures to our system. AuXTran obtains the substitution and then carries out its analysis to assemble the set of global constraints. The resulting generalised transformation is then added to the repertoire of OpTiSB and used within the opportunistic framework.

The generalisation procedure is not to be seen as the only possible one. Some refinements and adaptations when dealing with each syntactic category would allow us to obtain slightly different transformations. Whether the obtained transformations are more or less useful is a direct function of the generalisation procedure employed: if nothing is done, the obtained transformations would be particular to the pair of sample programs. More sophisticated generalisation methods can be used in order to enlarge the class of programs that could profit from the extracted transformations.

We would like to further elaborate on the following discussion topics:

- *Generalisation of Schemata* — Our proposed generalisation procedure is performed in a disciplined fashion so as not to under- nor over-generalise the schemata. This is achieved by the careful examination of each schematic construct and its context – for example, actual predicate symbols are not relevant if they appear both in the input and output schemata; references to user-defined predicates within the schemata should be replaced by generic literals and the definition of the predicate incorporated into the set of global constraints (e.g., *append/3* in our working example above); sequences of literals and terms are appropriately generalised if they play no important role in the transformation (i.e., they are not altered), and, depending on where they appear in the schema, may give rise to *length* constraints, and so on.
  We have thought, however, of ways to make the generalisation process more flexible. The first possibility would be to enable an expert programmer to interfere in the analysis of the substitution pairs and their translation to a global constraint – (s)he would decide on what parts of the schemata should be generalised and how this could be achieved (this is suggested in [11], in another context). Another alternative, which also relies on the expertise of a human, would be to offer a means to "tag" portions of the original programs as a protection against over-generalisation: those tagged constructs would be preserved unaltered (e.g., the occurrence and position of a cut, in some circumstances, should be maintained).
- *Correctness of Extracted Transformations* — As pointed out in the introduction above, a necessary condition that program transformations must meet is that they should be correct. Within the framework established in this article, we have suggested different alternatives to ensure that the extracted transformations are correct. One approach would operate on the object level, that is, on the pair of sample programs. We could initially prove that these two programs are equivalent in some kind of semantics and then go on to prove that the generalisation procedure does not change this status. In order to prove that two programs are equivalent in a *declarative semantics* [1], it is enough to show that they can be mapped using a finite sequence of applications of fold and unfold rules [10]. This can be done manually or automatically, exhaustively trying all possible applications of fold and unfold rules, in different orderings.
  An alternative approach would work on the level of the final generalised transformation schemata extracted. We suggest employing the *abstract* fold

and unfold rules proposed in [8] as a means to prove the correctness of the transformation. If an appropriate sequence of application of such rules is found then the transformation schema would be proven correct with respect to the procedural semantics of Prolog. The generalisation process would not need further consideration.

– *Need for Automating the Preparation of Transformations* — Our proposal hinges on the assumption that there might be a continuous need for new program transformations and since the process of devising these is error-prone, (semi-) automating would be useful. However, there is another approach to this issue: only a small repertoire of very expressive transformations is actually needed; it is prepared and proved correct once and for all, and nothing else will ever need to be added to the set of transformations [6].

References

1. Krzysztof R. Apt. *From Logic Programming to Prolog*. Prentice-Hall, U.K., 1997.
2. E. Chasseur and Y. Deville. Logic Program Schemas, Constraints and Semi-Unification. In *LNCS, Vol. 1463*. Springer, 1998.
3. N. E. Fuchs and M. P. J Fromherz. Schema-Based Transformations of Logic Programs. In *Proc. of LoPSTr'91*. Springer, 1992.
4. T. S. Gegg-Harrison. Exploiting Program Schemata in a Prolog Tutoring System. Technical Report CS-1993-11, Department of Computer Science, Duke University; Durham, North Carolina, U.S.A., April 1993. Reformatted version of PhD dissertation with the same title and equivalent content.
5. G. Huet and B. Lang. Proving and Applying Program Transformations Expressed with Second-Order Patterns. *Acta Informatica*, 11:31–55, 1978.
6. H. Büyükyıldız and P. Flener. Generalized Logic Program Transformation Schemas. In *LNCS, Vol. 1463*. Springer, 1998.
7. M. Proietti and A. Pettorossi. Transformations of Logic Programs: Foundations and Techniques. *J. Logic Progr.*, 19, 20:261–320, 1994.
8. J. D. C. Richardson and N. E. Fuchs. Development of Correct Transformation Schemata for Prolog Programs. In *LNCS, Vol. 1463*. Springer, 1998.
9. L. Sterling and E. Shapiro. *The Art of Prolog*. MIT Press, 1986.
10. H. Tamaki and T. Sato. Unfold/fold Transformations of Logic Programs. In *Proc. 2nd Int'l Conf. on Logic Programming*, Uppsala, Sweden, 1984.
11. W. W. Vasconcelos. *Extracting, Organising, Designing and Reusing Prolog Programming Techniques*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, August 1995.
12. W. W. Vasconcelos, M. A. Aragão, and N. E. Fuchs. Automatic Bottom-Up Analysis and Transformation of Logic Programs. In *LNAI, Vol. 1159*. Springer, 1996.
13. W. W. Vasconcelos and N. E. Fuchs. An Opportunistic Approach for Logic Program Analysis and Optimisation using Enhanced Schema-Based Transformations. In *LNCS, Vol. 1048*. Springer, 1996.
14. W. W. Vasconcelos and E. Meneses. A Pratical Approach for Logic Program Analysis and Transformation. In *Proc. of MICAI'2000*. Springer, 2000.

Modifications to the Credit Apportionment Mechanism of a Simple Classifier System

Carlos Zozaya-Gorostiza[1] and David R. Orellana-Moyao[2]

Instituto Tecnológico Autónomo de México, División Académica de Ingeniería
Río Hondo # 1, México D.F. 01000, México
[1]zozaya@lamport.rhon.itam.mx
[2]dorell@sisisa.podernet.com.mx

Abstract. This article describes several modifications performed to the credit assignment mechanism of Goldberg's Simple Classifier System [4] and the results obtained when solving a problem that requires the formation of classifier chains. The first set of these modifications included changes to the formula used to compute the *effective bid* of a classifier by taking into consideration the reputation of the classifier and the maximum bid of the previous auction in which a classifier was active. Noise was made proportional to the strength of the classifier and specificity was incorporated as an additional term in the formula that is independent from the bid coefficient. A second set of changes was related to the manner in which classifiers belonging to a chain may receive a *payoff* or a penalty from the environment in addition to the payments obtained from succeeding classifiers. We also tested the effect that *bridge classifiers* [13] have in the solution of the example problem by allowing the creation of shorter chains. Some experiments in which classifiers were better informed gave better results than those in which only the noise and the specificity were included in the computation of the effective bid.

Keywords. Classifier Systems, Credit Apportionment, Bucket Brigade, Classifier Sequences.

1 Introduction

The Bucket Brigade (BB) algorithm has been widely used as a learning procedure for Holland-type classifier systems (CS) such as Riolo's CFS-C [13]. This algorithm has the virtue of distributing payoff in a classifier chain by means of local transactions among the classifiers. No global knowledge is required and the system is capable of establishing hierarchies among classifiers, which are important for solving problems in large solution spaces. However, the BB presents several problems for generating and maintaining long chains of classifiers [21]. On one hand, the BB takes a long time to distribute payoff to those classifiers that are at the beginning of the chain; on the other, the first members of the chain end up having less steady-state strength than those at the end and the chain might become unstable. In a chain, the first classifiers of the chain depend upon the last ones to get their payoff and, at the same time, the latter require the former to "set-up" the stage before they can be activated.

In this paper we describe some modifications to the credit apportionment module of the Simple Classifier System (SCS) created by Goldberg [4]. Although this

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 235—246, 2000.

classifier system does not use a complete version of the BB, it exhibits behaviors that are common to other CS and allowed us to explore the effects of the proposed modifications in a more controlled environment with fewer degrees of freedom. Our results with a planning problem involving 78 classifiers (the Towers of Hanoi with three disks) show that some changes in the computation of the effective bid of classifiers give better results for learning sequences of classifiers in this problem and might be helpful for further research in this area.

2 Review of Previous Work on Credit Apportionment in CS

Wilson and Goldberg [21] published an excellent review of Classifier Systems literature up to 1989. Their paper described the main contributions that had been obtained in this field by researchers adopting two contrasting (but complementary) approaches: a) the *Michigan* approach in which apportionment of credit is implemented at the level of individual rules; and b) the *Pitt* approach in which entire rule sets are evaluated. Here we briefly revisit some of these contributions, paying special attention to those findings and recommendations aimed at improving the behavior of the BB.

Holland [8] discussed a list of relevant theoretical questions for CS, identified features that are shared by all adaptive nonlinear networks and provided multiple suggestions for extending the standard definitions of classifier systems. Some possible extensions include to change the size of the message list depending upon the urgency of the situation, to revise the definition of the environment by incorporating persistence and overlap of input messages and to use triggered genetic operators.

Riolo [13] analyzed the performance of the BB for creating long sequences of classifiers in a system called CFS-C. He explored using bridge classifiers that relocate strength to the beginning of the chain or to all the classifiers in the sequence. The first case was implemented by inserting a bridge classifier that was activated at the beginning of the chain and then remained active while the chain was being created. The second alternative used multiple classifiers that posted a message from each intermediate step. In the first case, strength was given directly to the classifier that initially activated the bridge; in the second it was redistributed to all those in the chain, but paying more to those at the beginning of the sequence.

Riolo [14] also studied the effects of the BB for the creation of classifier hierarchies in which exception-covering rules protect more general default rules from committing mistakes. He found that the strength of a default rule tends to rise until the exception rule can no longer protect it, because the steady state strength is inversely proportional to the specificity of the classifier. To solve this problem, he suggested to make the effective bid proportional to a power of the specificity.

Westerdale [17] demonstrated that a penalty scheme makes sense in the BB to introduce an incentive for a production to take itself out of the eligibility set. He proposed an altruist BB which instead of penalizing productions inside the eligibility set, rewards productions that are in the same locus (a set of productions that subsumes the eligibility set).

Grefenstette [6] developed a system that included two levels of learning: a Problem Solving level based on Riolo's CFS-C system in which strength was distributed using the BB, and a Learning level in which he used a GA on populations of structures

representing sets of 20 rules. He [7] also co-developed a system with a new algorithm for credit apportionment that rewarded sequences of classifiers. In this system, a *match set* consisted of all the rules with the highest match score (1 for each condition that is satisfied). Each action received a bid equal to the strength of the strongest rule that specified the action in its RHS (in contrast to Riolo's CSF-C where all are added). Then an action was selected with a probability proportional to its bid and inconsistencies were solved (rules whose action was different from the selected action were removed). The strengths of the remaining rules (i.e., the active rules) were adjusted at the end of an episode (when a sequence of rules was completed). The adjustment of rule strengths, called the *Profit Sharing Plan* (PSP), consisted of subtracting a fraction of the rule strength and adding the same fraction of the payoff.

Westerdale [18] compared the BB with the PSP of Grefenstette concluding that by using "cash balances" (strengths) as pre-computed averages of values of production sequences, the BBA reduces these biases. However, he found that there were cases in which any of the two approaches might be better and others in which the PSP worked best because the pure BBA was biased.

Wilson and Goldberg [21] suggested keeping chains short by promoting modularity in the BB. In this scheme, action sequences would be decomposed into modules at lower levels of a hierarchy. They recommended using an auction in which noise is changed adaptively depending on the variance of received payoffs.

Huang [9] implemented a context-array BBA that separated the space of all possible contexts (circumstances in which rules can fire) and employed array-valued bids (strengths) to estimate rule usefulness under different context subsets.

Compiani et al developed a system that learns letter sequences [3] and found that having a relative small message list size helps competition and learning, but might lead to misbehavior of the system.

Yates and Farley [22] tested four credit apportionment alternatives which differ from the BBA in respect to either inter-rule payment, environmental payoff, or both, to fix rule strength distortion when duplicates, subsumes or equivalent rules exist. They found that the alternative of dividing payoff in proportion to the bids made had characteristics which facilitate implicit discovery and the effective removal of equivalent rules.

Twardowski [15] compared four reinforcement learning algorithms for credit assignment: Backward Averaging (BA), the BB, a hybrid BB-BA, and a technique (1 step Q-learning) similar to the BB but that uses the maximum strength of all classifiers active in the previous step to compute payments. He found that the hybrid BB-BA gave the best results in a pole-balancing problem.

Wilson developed a simplified "zeroth-level" classifier system called ZCS [19]. In this system, credit assignment is similar to the BB with three exceptions: classifier specificities are ignored, there is no bid competition, and the quantity passed to preceding classifiers is reduced by a factor at each step. He found that, due to local payoffs and incomplete exploration, the system tends toward "path habits" that are suboptimal providing additional evidence of the challenge involving optimal long sequences of classifiers. Wilson also showed that the bucket brigade can behave like Q-learning [16] if the action with maximum strength of each cycle is used to update the strength of preceding classifiers.

Wilson later proposed to estimate the fitness of a classifier based on the accuracy of its predictions rather than on its strength [20]. He developed a system called XCS

that separates both concepts and that executes the genetic algorithm only in niches defined by the match sets. As a result, the combination of accuracy-based fitness with a niche GA provides XCS with the capability of developing a complete and accurate mapping from inputs and actions to payoff predictions. More recently, Barry identified and proposed mechanisms for solving a subclass of the "aliasing" problem that occurs in XCS in non-Markovian environments that provide the same message for two states in environmental positions having different payoffs [1, 2].

3 Modifications to the Credit Apportionment Mechanism

Goldberg's SCS [5] is a simple classifier system that includes the three components of any CS: a rule and message module, a credit apportionment mechanism and a genetic algorithm. However, its message list includes only one message and only a single classifier is activated at each *auction cycle.* This classifier is chosen by selecting the classifier with the highest *effective bid* (EB), which is computed (see Equation 1) as a product of a bid coefficient (C_{bid}), a linear function of the *specificity* of the classifier and its *strength* (S). A Gaussian noise $N(0,\sigma_{bid})$ is added to avoid biasing the results towards the status quo.

$$EB_i \leftarrow C_{bid} \, (e_{bid1} + e_{bid2} \, \text{specificity}_i) \, S_i + N(0,\sigma_{bid}) \quad (1)$$

In our research we tested the effects of several changes in the formula of the EB, as well as other modifications associated with the distribution of rewards *along chains* of classifiers in a sample problem.

3.1 Bid Modifications

The EB of a classifier was computed using Equations 2 and 3, where X_1 to X_4 are binary variables representing whether the corresponding term is included (Xi = 1) or not (Xi = 0) in the computation of the bid (see Section 4.2), *reputation* is an attribute that increases when a classifier is activated or contributes to an improved solution to the problem, and $EBmaxprev_i$ is the winning bid of the previous auction in which the classifier participated.

$$EB_i \leftarrow (C_{bid} + X_1 \, N(0,\sigma_{bid}) + X_2 \, \text{specificity}_i + X_3 \, / \, \text{reputation}_i) \, S_i \quad (2)$$

$$\text{IF } X_4 = 1 \text{ and } EB_i > Ebmaxprev_i \text{ THEN} \quad (3)$$
$$EB_i \leftarrow (EB_i + Ebmaxprev_i) \, / \, 2$$

We can explain the changes performed to the original formulation of the SCS as follows:

- In the SCS, the noise included in the computation of the EB is independent from the strength of the classifier. In the new formulation, noise is *proportional* to the strength of the classifier because the Gaussian value multiplies S_i. This also causes noise to play a more important role in EB calculation whenever $S_i > 1$.

- In the new formulation, the effect of *specificity* is independent of the bid coefficient (C_{bid}). One can achieve the same result in the original formula only for the particular case in which e_{bid2} is equal to $(1 / C_{bid})$.
- People, organizations and products have a *reputation* associated with them. In our experiments, we also wanted to include a parameter that would represent how good a classifier has been historically. We computed an attribute that is proportional to the number of cycles that the classifier has been active and to the number chains in which it has participated that improve the best solution thus far encountered (see Eq. 4). Initially, we thought to bias the selection of a classifier in favor of those with higher reputation. However, some experiments showed that reputation could lead to the creation of monopolies in which the same set of reputable classifiers repeatedly wins the auctions. Therefore, we decided to do the opposite in order to provide the CS with more exploratory power.

$$\text{reputation}_i \leftarrow (\text{wincount}_i + 1)\ (\text{bettercount}_i + 1) \qquad (4)$$

- In the SCS, classifiers bid part of their strength without having any knowledge of how much they would need to bid in order to be successful in such a task. In real life, people that bid take into consideration not only the amount they are willing to pay for something, but also the amount they consider to be *enough* for achieving what they want. We decided to keep each classifier informed by storing the value of maximum bid of the previous auction in which the classifier participated. In some experiments we adjust the EB by taking the average of the current EB and this previous maximum bid.

3.2 Payoff Modifications

In the BB, every time a classifier wins an auction, it pays its *bid* (not its EB) to those classifiers that inserted the messages that allowed for its activation. A classifier might also receive a payoff from the environment. However, in applications whose solution requires the creation of classifier chains, payoff is normally obtained until the last classifier of the sequence is fired. We tested the following modifications associated with the computation and distribution of payoff in the SCS:

- *Payoff to the last classifier or to all the chain.* As mentioned earlier, the SCS activates only one classifier per auction cycle. Therefore, it is easy to store the sequence of classifiers that has been obtained since the beginning of a trial (i.e., and intent to solve the problem). In our CS we let the system to create long chains and, if the desired solution is obtained, we give a reward to either all the classifiers of the chain or only to the last classifier. Before distributing the payoff, however, repeated classifiers are removed from the chain in order to eliminate loops.
- *Equal or proportional payoff.* In our experiments we tested giving the same reward to all recipients of the payoff or to reward them depending upon the position of each classifier in the chain (see Eq. 5). This formulation is different to the PSP scheme proposed by Grefenstette et al [7]. In the PSP, the system adjusts (up or down) the strengths of the classifiers in the chain to the level of payoff of the environment by adding a fraction of the difference between Payoff and S_i. Our scheme merely adds a reward that recognizes their contribution for "setting up the stage".

$$\text{Reward}_i \leftarrow \text{Payoff} * \text{Position}_i / (\text{Length of Chain} + 1) \quad (5)$$

- *Bonus for Improvement.* In problems where the chain of classifiers can be mapped to a sequence of operators that transform an initial state into another one (like the Towers of Hanoi), the length of any solution chain constitutes an upper bound to the best plan that can be obtained. If the new chain is shorter than the best solution that has been obtained so far, a bonus for improvement proportional to the improvement may be given to the classifiers that compose the new chain in order to reinforce their strengths.
- *Penalty.* Conversely, a penalty could be given to new chains if they are worse than the best solution that has been obtained. If the new chain is shorter than the best solution already found penalty may be introduced in order to reduce the strength of the classifiers in the chain strengths.

The first two of these modifications are intended to create incentives for *cooperation* among all the classifiers that participate in the generation of a chain by distributing rewarding these classifiers regardless of the quality of the final solution. The last two changes consider the *effectiveness* of the classifier chain by comparing new and previous solutions to the problem. This is similar to the behavior of the SPHINcsX system developed by Richards [12] that only gives the payoff when the new solution is better than one obtained previously.

3.3 Bridge Classifiers

Another modification was the insertion of *bridge classifiers* similar to those implemented by Riolo [13]. Since the SCS does not allow the activation of multiple classifiers at each auction cycle, we could not implement a classifier that would remain active during the generation of a chain. Therefore, bridges were used to create shorter chains from the initial to the desired state in order to distribute strength to all the classifiers that belong to these sequences.

4 Example Problem and Results

The application that was selected was the well-known Hanoi-3 problem of moving three disks from one position to another, say from pole 1 to pole 3, without putting a bigger disk on top of a smaller one. In this application, we wanted the classifier system to learn a sequence of valid movements (i.e., a plan) that modifies the initial state into the desired one by repeatedly solving the problem. Our goal was not to compete with a planning system that is more suited to find non-linear operator plans [23] but to have an interesting example in which we could test our modifications to the credit apportionment mechanism of the SCS.

4.1 Problem Representation

Each classifier was considered to be an operator that transforms a state of the world into another. The condition portion of a classifier represents the state when the operator is applied and the effector part indicates the new state that is obtained after the classifier is executed. A state was represented using a 3-position array corresponding to the small, medium and big disks respectively. The value in each position indicates the pole in which the corresponding disk is located. For instance, state 111 indicates that all disks are located in the first pole.

Figure 1 shows an example of two states (211 and 311) that can be obtained by performing one operator from the initial state. These operators (classifiers 111:211 and 111:311) represent the operations of moving the small disk from pole 1 to pole 2 or pole 3 respectively.

A complete solution space of the problem includes 27 states and 78 classifiers [11]. The optimal plan for going from state 111 to state 333 is a chain of the 7 classifiers that create the following sequence of states: 111, 311, 321,221, 223, 123, 133, 333. A bridge classifier represents an artificial operator that jumps from one state to another that is several operators closer to the desired state (for instance 321:123).

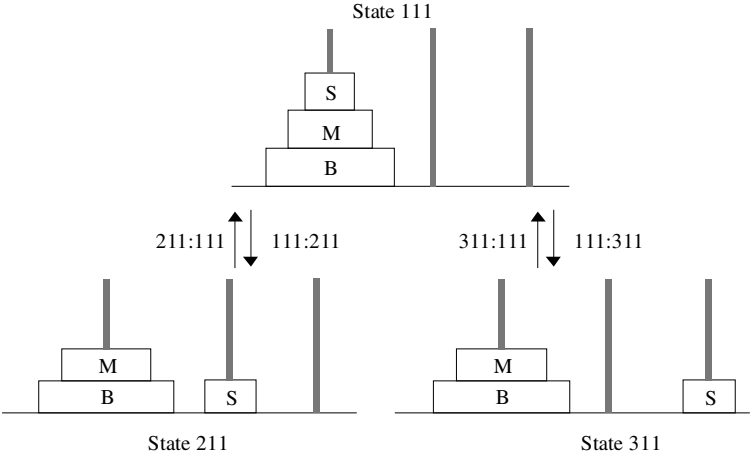


Fig. 1. Classifiers represent operators that transform a state into another

4.2 Experiments

Each experiment consisted of a combination of the proposed modifications to the SCS described. An 8-position binary array was describe the experiment: The first four positions of this array were associated with the computation of the effective bid (noise, specificity, reputation, and average with previous maximum bid) and a 1 represented that the corresponding modification would be included in the experiment. In position 5, a 1 represented that all the chain would get the reward, and a 0 that only the last classifier would receive it. In position 6, a 1 represented that the amount of reward would be proportional depending on the position of the classifier, and a 0 that

it would be equal. Finally a 1 in positions 7 and 8 represented that bonus and penalty would be included. Thus, the operation of a classifier system with only noise and specificity being considered is represented by array 1100:0000.

In order to compare the results obtained by the different experiments, the following variables were computed for each experiment:

- A variable called s*tability* represents the proportion of the most frequent chain in a certain number of chains produced by the system (for instance the last 20 trials), all of which are a valid solution to the problem. If stability is close to 1, the system has learned a path from the initial to the final state and/or is unable to obtain new solutions to the problem. Conversely, if stability is close to 0, the system seems to lack convergence.
- The proportion of classifiers that were able to win an auction during the execution of the experiment was labeled the *participation* of the classifiers. If participation is close to 1, the system was able to explore a considerable portion of the solution space because most classifiers were incorporated in chains and therefore new points in the state tree were obtained. Conversely, if participation is close to 0 a minor portion of the classifiers seem to dominate the auctions and the system seems to lack the power to explore a good proportion of the solution space.
- The *number of improvements* represents how many times a shorter run was obtained in comparison to the best solution thus far obtained.
- The *best chain* is the length of the shortest chain obtained by the system during the experiment. We are interested in short chains of classifiers that would transform the initial state into the desired one.
- Finally, the *stable length* is the length of the most frequent chain that the system is able to maintain during the final iterations of the experiment (the same period in which stability is computed).

Also, an overall strength for the experiment was computed using the formula of Eq. 6. A combination of modifications was good if it provided a stable length close to the optimal length.

$$\text{Strength} \leftarrow (\text{Optimal Length} / \text{Stable Length}) * \text{Stability} + \text{Participation} \quad (6)$$

4.3 Results of Experiments with Deterministic Bids

We ran only one outermost cycle per experiment with 100 trials each. Table 1 shows the results for some of these experiments ranked with respect to their strength. For instance, in experiment 0111:0100 a stable solution of 7 movements, which is optimal, was obtained. In this experiment, 91% of the classifiers were fired at least once and the system improved the solution three times (with respect to the best solution obtained so far). The Table shows that bid modifications determined the overall results of the experiment, regardless of the combinations related to the distribution of payoff. The best results were obtained when reputation and specificity were included in the bid process.

Table 1. Some Deterministic Experiments without Bridge Classifiers

Experiment	Min. Length	Stable Value	Stability	Improv.	Particip.	Strength
0110:0100	7	7	0.91	3	0.97	1.88
0111:0100	7	7	0.91	3	0.97	1.88
0010:0100	7	7	0.77	4	0.97	1.74
0101:0111	11	11	0.99	1	0.31	0.94
0001:1100	11	11	0.97	1	0.31	0.92

Figure 2 shows the first trials of experiment 0111:0100 in which reputation, specificity and average bid were incorporated. At the beginning of the run, the system finds several long chains but after 11 trials it reaches a stable solution of 7 steps. This behavior was also observed in experiments whose final strength was from 1.76 to 1.88. Other experiments, such as those in which payoff was divided among all the classifiers of the chain lasted longer in reaching a stable solution. For instance, experiment 0010:0100 obtained a stable value after 25 trials. Other experiments, such as 0001:1100, reached a non-optimal stable value.

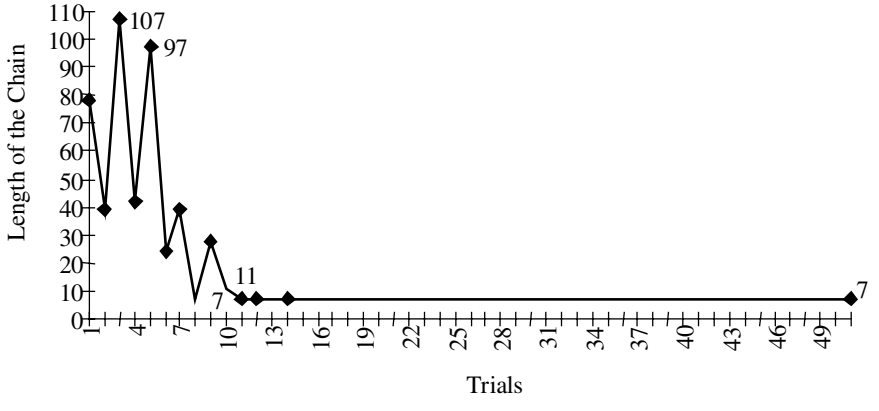


Fig. 2. Run of Experiment 0111:0100

When bridge classifiers were included, the system behaved in a similar manner. The best results were obtained when reputation, specificity and the maximum previous bid were taken into consideration for computing the effective bids regardless of the payoff scheme. In some of these runs the system first learns an optimum (but artificial) path of 5 movements and then adjusts this solution to the 7-step chain after the bridge classifiers are removed. In other cases (such as experiment 0011:1111) the system was unable to find a solution after the bridges were removed.

4.4 Results of Experiments with Stochastic Bids

A formal comparison of these experiments using the Duncan test of significant ranges [10 concluded that some experiments were statistically better than others with a 95% confidence (see [11] for details). Experiment 1100:0000, corresponding to a basic classifier system with noise and specificity (although our handling of these two

elements is somehow different as explained previously), gave better results than other experiments that included bonus, penalties and payoff distribution. Again, results were better when classifiers were more informed. However, these experiments were more sensitive to the modifications of the payoff than those in which bids were deterministic. Also, due to the Gaussian noise, the system achieved stability in a longer number of trials.

5. Conclusions

This article described two sets of modifications performed to the bidding and payoff distribution schemes of Goldberg's SCS [4]. Modifications to the bidding process were implemented by changing the formula used to compute the *effective bid* (EB) of a classifier during each auction cycle. Modifications to the *payoff distribution* scheme were aimed at creating incentives for cooperation among the classifiers that belong to a successful sequence by giving them a reward that complements the payments received from other classifiers. Both changes created a hybrid scheme in which the BB is used during the generation of classifier sequences but the strengths can be adjusted when a sequence is finished.

The Hanoi-3 problem provided a useful example that allowed the creation of very long chains of more than 500 classifiers. In this example, however, the modified SCS was able to quickly converge to short and, in most cases, optimal sequences. Those experiments in which classifiers used information about their reputation and/or the maximum bid of a previous auction for computing their effective bids gave better results than those in which only the noise and the specificity were considered.

Certainly, no claim is made about the generality of these results at this point, since they could be dependent upon factors related to the particular Hanoi-3 problem (e.g., the shape of the solution space). However, our experiments provide evidence that storing information in the classifiers to complement their local decision processes during bidding constitutes a research task worth exploring.

Our work provides many opportunities for further research. An interesting extension would be to test the effects of our modifications in a more complex CS that allows for the activation of multiple classifiers at each auction cycle. Another alternative would be to develop experiments in which each modification can be partially included (i.e., the X variables are redefined to include real values between 0 and 1). Further testing of our modifications in more complex problem domains, particularly in non-Markovian environments is also needed. Finally, since we did not use a genetic algorithm to generate new populations of classifiers, this constitutes another opportunity for further research in which our modifications to the credit apportionment mechanism of SCS could be incorporated into other systems such as ZCS [19] or XCS [20].

References

1. Barry, A. *Aliasing in XCS and the Consecutive State Problem: 1 - Effects* In *Proceedings of the Genetic and Evolutionary Computation Conference GECO'99*. Banzhaf, W., *et al.*, eds.: Morgan Kauffman (1999) 19-26
2. Barry, A. *Aliasing in XCS and the Consecutive State Problem: 2 - Solutions* In *Proceedings of the Genetic and Evolutionary Computation Conference GECO'99*. Banzhaf, W., *et al.*, eds.: Morgan Kauffman (1999) 27-34
3. Compiani, M., Montanari, D., and Serra, R. *Asymptotic Dynamics of Classifier Systems* In *Proceedings of the Third International Conference on Genetic Algorithms*. Schaeffer, J.D., ed.: Morgan Kaufmann (1989) 298-303
4. Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA.: Addison-Wesley (1989)
5. Goldberg, D.E., *Probability Matching, the Magnitude of Reinforcement, and Classifier System Bidding.* Machine Learning. Vol. 5, (1990) 407-425
6. Grefenstette, J.J. *Multilevel Credit Assignment in a Genetic Learning System* In *Proceedings of the Second International Conference on Genetic Algorithms*. Grefenstette, J.J., ed.: Lawerence Erlbaum Assoc. (1987) 202-209
7. Grefenstette, J.J., Ramsey, C.L., and Schultz, A.C., *Learning Sequential Decision Rules Using Simulation Models and Competition.* Machine Learning. Vol. 5, (1990) 355-381
8. Holland, J.H. *Genetic Algorithms and Classifier Systems: Foundations and Future Directions* In *Proceedings of the Second International Conference on Genetic Algorithms*. Grefenstette, J.J., ed.: Lawerence Erlbaum Assoc. (1987) 82-89
9. Huang, D., *A Framework for the Credit-Apportionment Process in Rule-Based Systems.* IEEE Transactions on Systems, Man and Cybernetics. Vol. 19, No. 3 (1989) 289-4998
10. Miller, I. and Freund, J., *Probability and Statistics for Engineers*: Prentice Hall (1986)
11. Orellana-Moyao, D.R., *Modificaciones al Mecanismo de Asignación de Crédito de un Sistema Clasificador Básico*, Instituto Tecnológico Autónomo de México: México, D.F. (1998)
12. Richards, R.A., *Zeroth-order Shape Optimization Utilizing a Learning Classifier System*, in *Mechanical Engineering Department*, Stanford University (1995)
13. Riolo, R.L. *Bucket Brigade Performance: I. Long Sequences of Classifiers* In *Proceedings of the Second International Conference on Genetic Algorithms*. Grefenstette, J.J., ed.: Lawerence Erlbaum Assoc. (1987) 184-195
14. Riolo, R.L. *Bucket Brigade Performance: II. Default Hierarchies* In *Proceedings of the Second International Conference on Genetic Algorithms*. Grefenstette, J.J., ed.: Lawerence Erlbaum Assoc. (1987) 196-201
15. Twardowski, K. *Credit Assignment for Pole Balancing with Learning Classifier Systems* In *Proceedings of the Fifth International Conference on Genetic Algorithms*. Forrest, S., ed. (1993) 238-245
16. Watkins, C.J.C.H., *Learning from Delayed Rewards*, Cambridge University: Cambridge, UK. (1989)
17. Westerdale, T.H. *Altruism in the Bucket Brigade* In *Proceedings of the Second International Conference on Genetic Algorithms*. Grefenstette, J.J., ed.: Lawerence Erlbaum Assoc. (1987) 22-26
18. Westerdale, T.H. *A Defense of the Bucket Brigade* In *Proceedings of the Third International Conference on Genetic Algorithms.* Schaeffer, J.D., ed.: Morgan Kaufmann (1989) 282-290
19. Wilson, S.W., *ZCS: A Zeroth Level Classifier System.* Evolutionary Computation. Vol. 2, No. 1 (1994) 1-18

20. Wilson, S.W., *Classifier Fitness Based on Accuracy.* Evolutionary Computation. Vol. 3, No. 2 (1995) 149-175
21. Wilson, S.W. and Goldberg, D.E. *A Critical Review of Classifier Systems* In *Proceedings of the Third International Conference on Genetic Algorithms*. Schaeffer, J.D., ed.: Morgan Kaufmann (1989) 244-255
22. Yates, D.F. and Farley, A. *An Investigation into Possible Causes of, and Solutions to, Rule Strength Distortion Due to the Bucket Brigade Algorithm* In *Proceedings of the Fifth International Conference on Genetic Algorithms*. Forrest, S., ed.: Morgan Kaufmann (1993) 246-253
23. Zozaya-Gorostiza, C. and Díaz-Infante, E., *Planning by Merging Totally Ordered Object Plans.* Journal of Expert Systems with Applications. Vol. 14, No. 1/2 (1998)
24. Zozaya-Gorostiza, C. and Estrada, L.F., *Incorporating Heuristics and a Meta-Architecture in a Genetic Algorithm for Harness Design*, in *Advances in Formal Design Methods for Computer Aided Design*, Gero, J. and Sudweeks, F., Editors, IFIP WG5.2. (1994)

A Framework for Interleaving Planning-while-Learning and Execution

Marcello Balduccini

Centro di Ricerca "Informatica Interattiva"
Università degli Studi dell'Insubria
via Ravasi 2, I-21100 Varese (Italy)
ph:+39-332-250212 fax:+39-332-281308
marcy@mail.varbio.unimi.it

Abstract. Interacting with the environment in presence of incomplete information requires the ability to acquire new knowledge from the interaction with the environment and to employ it when deliberating about which actions to execute. The ability to identify a particular environmental behaviour by inspecting perceptual feedback greatly contributes to completing the knowledge available to the agent. This paper introduces a formal framework for interleaving planning-while-learning and execution in partially specified environments. Planning-while-learning combines conventional planning with the search of the environmental behaviour model that best fits the experienced behaviour of the environment. Heuristics for early termination of planning and assumptions are used in order to reduce the cost of planning. Sufficiency conditions are given that guarantee the soundness and the completeness of the agent's control system w.r.t. the environmental model and the goal.

1 Introduction

In the area of intelligent autonomous agents and of autonomous robots, many different approaches have been attempted to reconcile the need to carefully plan sequences of actions with the need to interact timely with a rapidly evolving environment.

Some researchers have proposed to substitute planning with more light-weighted reasoning techniques, like behaviours ([9], [8]), policies ([3], [4], [10], [2]) and others. The drawback of these solutions is that, while they solve the problem of a timely interaction with the environment, they pose new problems because of the general reduction in the representation and reasoning power.

On the other hand, other researchers have developed solutions which reduce the cost of planning, but do not affect the representation and reasoning ability. State of the art approaches ([11], [12], [1], [6], [7]) adopt early termination heuristics, assumptive planning, abstract planning, often combined together, to form a powerful reasoning device able to interact timely with the environment.

Even if it is a widely spread opinion that learning may increase the autonomy of both agents and robots by allowing them to refine their a-priori environmental

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 247–259, 2000.

model according to the empirical experience, examples of well formalized frameworks for agents integrating learning and planning while acting timely with the environment are rare ([7]).

The present paper tries to fill this lack by introducing a formal framework for interleaving planning-while-learning and execution in partially specified environments. In presence of incomplete information, interleaving ([5]) is a very efficient way to perform planning, since it allows to exploit the perceptual feedback from the environment in order to increase the knowledge available to the agent before a new planning phase is started, thus reducing its computational cost. Planning-while-learning ([14]) combines conventional planning with the search, within a given initial set, of the environmental behaviour model that best fits the experienced behaviour of the environment.

Our approach is based upon Nourbakhsh's work on the interleaving of planning and execution ([11], [12]) and Safra's and Tennenholtz's paper on planning-while-learning ([14]); particular attention is paid to the sufficiency conditions that guarantee the soundness and the completeness of the agent's control system w.r.t. the environmental model and the goal. Heuristics for early termination of planning and assumptions about both the current state of the world and the actual environmental behaviour are used in order to reduce the cost of the planning process.

The paper is structured as follows. In Sect. 2 we introduce our version of Nourbakhsh's framework for the interleaving of planning and execution, which represents the state of the art in this area. It will constitute the basis for Sect. 3, where we describe our framework for interleaving planning-while-learning with execution and deal with soundness and completeness. In Sect. 4 we compare our work with relevant research in this area. In Sect. 5 we draw the conclusions from our work and highlight possible future developments.

2 Interleaving Planning and Execution

With respect to modeling a robot's interaction with the environment, Nourbakhsh ([11]) has shown that the world may be viewed as a system made up of two distinct, interacting parts: the robot and its environment, which are connected only by the percept and the action streams.

In our framework, a *percept* is formally defined as a vector representing the instantaneous image of the robot's inputs. An *action* is a vector representing the instantaneous image of the robot's outputs.

The interaction between the agent and the environment is modeled as the interaction between two finite automata, the robot finite automaton and the environment finite automaton.

The **robot finite automaton** or *rfa* is defined as a tuple $< P, B, A, int, ext, b >$, where:

- P is a set of input objects (the agent's possible percepts);
- B is a set of internal states;

- A is the set of output objects (the agent's possible actions);
- *int* is a function $P \times B \rightarrow B$ that updates the robot's internal state;
- *ext* is a function $B \rightarrow A$ that prescribes the robot's output;
- b is a member of B (the robot's initial internal state).

The **environmental finite automaton** or *efa* is defined as a tuple $<A, S, P, do, see, s>$, where:

- A is the environment's set of input objects (the robot's actions);
- S is the environment's set of states;
- P is the environment's set of output objects (the robot's percepts);
- *do* is a function $A \times S \rightarrow S$ that updates the environment's state;
- *see* is a function $S \rightarrow$ P that updates the robot's percept;
- s is a member of S (the initial state of the world).

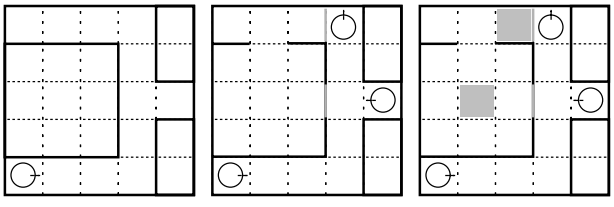


Fig. 1. (a) A simple maze. (b) An incompletely specified maze. (c) A maze with objects with incompletely specified behaviour.

Example 1. Suppose that our robot is put in the maze depicted in Fig. 1(a) at position $(1, 1)$. The robot has four binary inputs, corresponding to four wall detectors, placed one on each side of the robot. Three actions are available: *ahead*, *turn-left* and *turn-right*. The control system of the robot is programmed to go ahead whenever possible and to turn left otherwise. The $<rfa, efa>$ system for this setting may be described as follows:

- P is the set of all binary-valued vectors of length 4;
- $A = \{ahead, turn{-}left, turn{-}right\}$;
- S is the set of tuples $<maze{-}conf, pos>$ where $maze{-}conf$ is the configuration of the maze shown in Fig. 1(a) and *pos* is the position of the robot, ranging all over the maze, and its bearing (n,s,e,w);
- s is the tuple $<maze{-}conf, (1, 1, e)>$;
- $B = \{ahead{-}clear, ahead{-}blocked, start\}$ where *start* represents the robot's initial state, when nothing is known yet about the sensors' state;
- b is set to *start*;
- *do* updates the position of the robot according to the action performed;
- *see* returns the value of the input channels of the robot according to the position of the walls around the robot;

– *int* is defined as follows:

$$int(p, b) = \begin{cases} ahead{-}clear & \text{if ahead-bit of } p \text{ is } 0 \\ ahead{-}blocked & \text{if ahead-bit of } p \text{ is } 1 \end{cases};$$

– *ext* is defined as follows:

$$ext(b) = \begin{cases} ahead & \text{if } b = ahead{-}clear \\ turn{-}left & \text{if } b = ahead{-}blocked \end{cases}.$$

In many cases, the unreliability of sensors and effectors, as well as incomplete knowledge about the environment, prevent the designer from coming up with an *efa*, which by definition doesn't permit to express *incomplete information*.

For this reason we introduce a **partially specified environmental finite automaton** or *pefa*, which is defined as a tuple $< A, S, P, effect, sense, I >$, where:

– A, S and P are defined like in *efa*;
– *effect* is a function $A \times \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ that updates the environment's set of possible states;
– *sense* is a function $P \times \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ that returns the maximal subset of the second argument consistent with the robot's percept;
– I is a subset of $\mathcal{P}(S)$ (the set of possible initial states of the world).

The *pefa* makes it possible to express incomplete information by reasoning in terms of the set of the environment's *possible* states (the so-called *state-set*).

Example 2. Suppose that our robot is put in the maze depicted in Fig. 1(b). There is incomplete information about the initial position, which may be either $(1, 1)$ or $(5, 3)$ or $(4, 5)$, and about its initial bearing, like shown in figure. There is incomplete knowledge, as well, about the walls drawn in grey: only one of them exists, but it is not known which one. The other features of the environment are like in Example 1.

The *pefa* for this setting may be described as follows:

– P and A are defined like in Example 1;
– S is the set of tuples $<maze{-}conf, pos>$ where $maze{-}conf$ is one of the two possible configurations of the maze shown in Fig. 1(b) and *pos* is defined like in Example 1;
– I is the set of tuples $<maze{-}conf, pos>$, where $maze{-}conf$ is one of the two possible configurations of the world and *pos* is either $(1, 1, e)$ or $(5, 3, w)$ or $(4, 5, n)$;
– $effect(a, S')$ and $sense(p, S')$ are defined following the environment's specifications.

Intuitively, many *efa*'s are "compatible" with a given *pefa*. We formally define the notion of *consistency* as follows:

Definition 1. *A pefa* $< A, S, P, effect, sense, I >$ *is consistent with an efa* $< A, S, P, do, see, s >$ *iff:*

1. $\forall S' \subseteq S \;\; \forall p \in P \; sense(p, S') \supseteq \{\bar{s} \mid \bar{s} \in S' \wedge see(\bar{s}) = p\}$;
2. $\forall S' \subseteq S \;\; \forall a \in A \; effect(a, S') \supseteq \{\bar{s} \mid \exists s' \; (s' \in S' \wedge \bar{s} = do(a, s'))\}$;
3. $s \in I$.

Consistency of a *pefa* with an *efa* guarantees that a state-set tracker (see Definition 3), applied to an environment modeled by the *efa*, tracks the environment using the information provided by the *pefa*, despite its incompleteness.

Now we can define the planning problem in terms of a *pefa* and a goal. A *problem instance* is a tuple $<pefa, G>$ where *pefa* is defined as usual and G is a *set of state sets*, representing the disjunction of mutually exclusive goals, each expressed by a state set. The concept of solution of a planning problem is formalized by the following definitions.

Definition 2. *Satisfaction of a goal G by a state-set I is expressed by the relation $satisfies(I, G)$ defined upon $\mathcal{P}(S) \times \mathcal{P}(\mathcal{P}(S))$ as follows: $satisfies(I, G) \iff \exists g \; (g \in G \wedge I \subseteq g)$.*

Definition 3. *A* state set tracker (sst) *for a pefa $< A, S, P, effect, sense, I >$ and an action-percept history $\alpha = \{a_1, p_1, a_2, p_2, \ldots, a_n, p_n\}$ is a robot finite automaton $<P, B, A, int, ext, b_1>$ where: (1) $B = \mathcal{P}(S)$; (2) $int(p_i, b_i) = sense(p_i, effect(a_i, b_i))$; (3) $ext(b_i) = a_i$; (4) $b_1 = I$.*

Definition 4. *An rfa is a* solution *to a problem instance $<pefa, G>$ iff, for every efa consistent with pefa, given the action-percept history $\alpha = \{a_1, p_1, a_2, p_2, \ldots, a_n, p_n\}$ of the system $<rfa, efa>$, the state set tracker $sst(pefa, \alpha)$ in system $<sst, efa>$ computes b_n s.t. $satisfies(b_n, G)$ is true.*

In our approach we employ conditional planning to build plans leading to the goal[1]; search for solution plans is performed in the space of state-sets.

The interleaving of planning and execution is achieved by splitting the planning process in several planning episodes which are alternated to execution (see [11], [5], [13], [6], [1], [7])).

We divide the monolithic planning process into smaller planning episodes by acting along two dimensions:

- *early termination heuristics (ETH)* cause interruption of a planning episode *before* a solution plan is found (see [6], [7]);
- *assumptions* about the current state-set reduce the size of the search space, simplifying the planning process as well.

A great deal of early termination heuristics have been investigated by the researchers. We distinguish between early termination *rules*, which preserve the

[1] Conditional plans prescribe the action to be performed according to every history of percepts that can be encountered at execution time ([11]). For example, "*reach-elevator*; if *elevator-present* then *use-elevator* else *use-stairs*" is a fragment of a conditional plan for reaching the ground floor.

soundness and the completeness of the control system of the robot[2] (e.g. the viable plan termination rule and the forced plan termination rule in [11]), and early termination *heuristics*, which don't preserve them (e.g. the resource-based termination heuristics in [6]).

Assumptive planning is realized by means of the *state selection function* which is applied to the current state-set at the beginning of each planning episode.

Definition 5. *The* state selection function *is a function* $sel: \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ *used to select, within the given state-set, those states which are deemed relevant with respect to the current situation.*

A function sel is a state selection function *iff* $\forall I\,(I \subseteq S \Rightarrow sel(I) \subseteq I)$.

A general scheme of an *rfa* implementing interleaved planning and execution by means of both early termination heuristics and assumptive planning can be given in algorithmic form as follows:

Algorithm 1 (et-CSEL). *Given:*

- *a pefa* $<A, S, P, effect, sense, I>$;
- *a state of state-sets G (the goal of the agent);*
- *a state selection function sel;*
- *a function* et–plan *that, given the current state-set J, the current goal G and the effect and sense functions, returns a conditional plan which may be either a solution plan or a partial plan (if an early termination heuristic was applied);*
- *the relation* $satisfies$;
- *a function* first *which returns the first action of a conditional plan;*

compute:

1. *Let* $p =$ *current percept*
2. $I = sense(p, I)$
3. *if* $satisfies(I, G)$ *then terminate*
4. $J = sel(I)$
5. $CPlan =$ et–plan$(J, G, effect, sense)$
6. *Let* $a =$ first$(CPlan)$
7. $I = effect(a, I)$
8. $output = a$
9. *goto 1*

The algorithm performs continuous selection ([11]): assumptions and plans are recomputed from scratch after the execution of every action.

The drawback of this approach is that the total conditional planning cost may be much higher than in an algorithm in which replanning occurs only when strictly needed. However, in domains with incomplete information, in which the knowledge carried by the percepts plays a fundamental role, the tight interleaving of planning and execution imposed by Algorithm 1 can have significant benefits.

The planning algorithm implementing et–plan is specified as follows:

[2] w.r.t. the goal specified and the environmental model.

Algorithm 2 (et–plan). *Given:*

- *a state-set I representing the current state-set;*
- *a state of state-sets G representing the goal of the agent;*
- *the effect and sense functions;*
- *a relation* check–ETH *that checks whether early termination heuristics can be triggered;*
- *a function* nondet *that performs non-deterministic choice;*
- *the set A of possible actions and the set P of possible percepts;*

compute:

1. *if* $satisfies(I, G)$ *then return*
2. *if* check–ETH(I) *then return*
3. $a =$ nondet(A)
4. $I' = effect(a, I)$
5. *for every* $p \in P$ *do:*
6. $I'' = sense(p, I')$
7. et–plan$(I'', G, effect, sense)$

3 Interleaving Planning-while-Learning and Execution

The basic framework (the *pefa* framework) that we have presented so far makes it possible to cope with environments about which only incomplete information is available.

Three forms of incomplete information can be handled:

- partial knowledge about the initial state of the world (represented by the set I);
- partial knowledge about the behaviour of the environment (represented by the *effect* function);
- unreliability of sensory information (represented by the *sense* function).

In many cases, the performance of the agent might be improved by tuning the description of the environment's behaviour at *run-time*. This contrasts with the a-priori specification of the *effect* function given in the *pefa*.

The idea behind *planning-while-learning* is that the agent is given a set of possible behaviours of the environment before interaction starts; while interaction with the world progresses, the agent uses the information available through sensors to select, from the initial set, the behaviour which best fits the experienced behaviour of the environment.

In order to specify a set of possible environment's behaviours we introduce a **planning-while-learning environmental finite state automaton** or *plefa*, defined as a tuple $<A, S, P, E, update, sense, F, I>$ where:

- A, S, P, *sense*, I are defined like in *pefa*;
- $E \subseteq \mathcal{P}(S)^{A \times \mathcal{P}(S)}$ is the set of the possible behaviours of the environment;

- *update* is a function $\mathcal{P}(E) \times \mathcal{P}(S) \times A \times \mathcal{P}(S) \rightarrow \mathcal{P}(E)$ that updates the set of possible behaviours of the environment;
- $F \subseteq E$ is the set of the possible initial behaviours of the world;

Example 3. Suppose that our robot is put in the maze depicted in Fig. 1(c). Besides the specifications of Example 2 the cells with the grid can either prevent robot's crossing or make the robot advance of two cells instead of just one or have no special effect. The behaviour is unknown, but it is fixed over time and is the same for all the cells. The *plefa* for this setting may be described as follows:

- P, A, S, I and *sense* are defined like in Example 2;
- $E = F = \{b_1, b_2, b_3\}$, where b_i are the effect functions corresponding to the behaviours described above.

The definitions that follow describe the concept of solution to a problem instance and the relation among *plefa*, *pefa* and *efa*.

Definition 6. *The effect function* w.r.t. a set of behaviours F *is a function* $effect_F \colon A \times \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ *which describes the consequences of an action from a state-set according to* every *behaviour belonging to the given behaviour-set:* $\forall I \subseteq S \;\; \forall a \in A \;\; effect_F(a, I) = \bigcup \{I' \mid \exists b \; (b \in F \wedge b(a, I) = I')\}$.

Definition 7. *A* state-behaviour set tracker (sbt) *for a plefa* $<A, S, P, E, update, sense, F, I>$ *and an action-percept history* $\alpha = \{a_1, p_1, a_2, p_2, \ldots, p_n, a_n\}$ *is a robot finite automaton* $<P, B, A, int, ext, b_1>$ *where:*

- $B = \{<s, e> \mid s \in \mathcal{P}(S) \wedge e \in \mathcal{P}(E)\}$;
- $int(p_i, b_i) = <s_{i+1}, update(e_i, s_i, a_i, s_{i+1})>$, *where* $s_{i+1} = sense(p_i, effect_{e_i}(a_i, s_i))$ *and* $b_i = <s_i, e_i>$;
- $ext(b_i) = a_i$;
- $b_1 = I$.

Definition 8. *A plefa* $<A, S, P, E, update, sense, F, I>$ *is consistent with a pefa* $< A, S, P, effect, sense, I >$ *iff, for every efa consistent with pefa, the action-percept history* $\alpha = \{a_1, p_1, a_2, p_2, \ldots, a_n, p_n\}$ *of the system* $<rfa, efa>$ *makes the state-behaviour set tracker* $sbt(plefa, \alpha)$ *compute* $e_k = \{effect\}$ *for an index* $k \leq n$ *when applied to the system* $<sbt, efa>$.

Definition 9. *A plefa* $<A, S, P, E, update, sense, F, I>$ *is consistent with an efa* $<A, S, P, do, see, s>$ *iff there exists a pefa such that:*

1. *the plefa is consistent with the pefa, and*
2. *the pefa is consistent with the efa.*

Definition 10. *An rfa is a* solution *to a problem instance <plefa, G> iff, for every efa consistent with plefa, given the action-percept history* $\alpha = \{a_1, p_1, a_2, p_2, \ldots, a_n, p_n\}$ *of the system <rfa, efa>, the state-behaviour set tracker* $sbt(plefa, \alpha)$ *in system <sbt, efa> computes* s_n *s.t.* $satisfies(s_n, G)$ *is true.*

The conditional planning approach that we use in our extended framework (the *plefa* framework) performs a search in the space of the couples <*state-set*, *behaviour-set*>.

The fact that the behaviour-set is part of the search space results in being it possible for the agent to build plans which take into account also the information gained about the environment's behaviour: early identification of the actual behaviour is essential for a successful interaction with the world, since sequences of actions may exist which lead to the goal for some possible behaviours but make the goal unreachable for other possible behaviours.

We represent the evolution of the world during interaction by means of a **state-behaviour set graph**.

Definition 11. *A* state-behaviour *set graph is a directed bipartite graph with effectory nodes and perceptory nodes.*

An effectory node is a tuple <state-set, behaviour-set> *with action-labeled arcs leading away from it. Each action arc labeled with action a connects an effectory node* $<S_1, B_1>$ *to a perceptory node* $<S_2, B_1>$ *iff* $S_2 = effect_{B_1}(a, S_1)$ *where* $effect_{B_1}$ *is the effect function w.r.t.* B_1.

A perceptory node is a tuple <state-set, behaviour-set> *with percept-labeled arcs leading away from it. Each percept arc labeled with percept p connects a perceptory node* $<S_1, B_1>$ *to an effectory node* $<S_2, B_2>$ *iff* $sense(p, S_1) = S_2$.

An effectory node $<S_1, B_1>$ *is connected to another effectory node* $<S_3, B_2>$ *through a perceptory node* $<S_2, B_1>$ *iff* $B_2 = update(B_1, S_1, a, S_3)$ *where a is the action labeling the arc from* $<S_1, B_1>$ *to* $<S_2, B_1>$.

In the *plefa* framework the interleaving of planning-while-learning and execution is obtained by means of early termination heuristics and assumptions.

Assumptions can be made either about the current state-set, like in the *pefa* framework, or about the current behaviour-set. In fact, the current behaviour-set can be quite large and it can contain elements which are not likely to represent the actual behaviour of the environment.

Planning using a large behaviour-set can be very time consuming, therefore making appropriate assumptions about which behaviours are more likely to represent the actual environmental behaviour can result in a relevant performance improvement.

Definition 12. *The* behaviour selection function *is a function* $sel_b: \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ *used to select, among a set of possible behaviours, those which are considered relevant with respect to the current situation.*

A function sel_b *is a* behaviour selection function *iff* $\forall F(F \subseteq E \Rightarrow sel_b(F) \subseteq F)$.

We give now a simple algorithm for implementing interleaved planning-while-learning and execution by means of both early termination heuristics and assumptive planning:

Algorithm 3 (pl-CSEL). *Given:*

- *a plefa* $<A, S, P, E, update, sense, F, I>$*;*
- *a state of state-sets* G *(the goal of the agent);*
- *state and behaviour selection functions* sel *and* sel_b*;*
- *a function* et–plan_{pl} *that, given the current state-set* J*, the current behaviour-set* C*, the current goal* G *and the update and sense functions, returns a conditional plan;*
- *the relation* $satisfies$ *and the function* first *(see Algorithm 1);*

compute:

1. *Let* $p =$ *current percept*
2. $I = sense(p, I)$
3. *if* $satisfies(I, G)$ *then terminate*
4. $J = sel(I)$
5. $C = sel_b(F)$
6. $CPlan = \text{et–plan}_{pl}(J, C, G, update, sense)$
7. *Let* $a = \text{first}(CPlan)$
8. $I' = effect_F(a, I)$
9. $output = a$
10. *Let* $p =$ *current percept*
11. $I' = sense(p, I')$
12. $F = update(F, I, a, I')$
13. $I = I'$
14. *goto 3*

The planning algorithm implementing et–plan_{pl} is specified as follows:

Algorithm 4 (et–plan_{pl}). *Given:*

- *a state-set* I *representing the current state-set;*
- *a state-set* F *representing the current behaviour-set;*
- *a state of state-sets* G *representing the goal of the agent;*
- *the update and sense functions;*
- *a relation* check–ETH *and a function* nondet *(see Algorithm 2);*
- *the set* A *of possible actions and the set* P *of possible percepts;*

compute:

1. *if* $satisfies(I, G)$ *then return*
2. *if* check–ETH(I) *then return*
3. $a = \text{nondet}(A)$
4. $I' = effect_F(a, I)$
5. *for every* $p \in P$ *do*
6. $I'' = sense(p, I')$
7. $F' = update(F, I, a, I'')$
8. $\text{et–plan}_{pl}(I'', F, G, update, sense)$

3.1 Theoretical Properties of the *plefa* Framework

We briefly introduce the theoretical properties of the algorithms of the *plefa* framework, namely Algorithms 3 and 4. Sufficiency conditions for the soundness and the completeness of our *rfa* w.r.t. a given problem instance $<plefa, G>$ will be presented. Similar results regarding the *pefa* framework and the control system built upon Algorithms 1 and 2, together with proofs, can be found in [11].

Note that we restrict our attention to *semistatic environments*, where the world does not change sensibly during deliberation.

Definition 13. *An rfa is* sound *w.r.t. a problem instance <plefa, G> iff for each efa consistent with the plefa,* if *system <rfa, efa> terminates at time* t_n, then s_n *makes true* $satisfies(s_n, G)$, *where* s_n *is intended in the sense of Definition 7.*

Definition 14. *An rfa is* sound w.r.t. the *plefa* framework *iff, for any problem instance* Π, *the rfa is sound w.r.t.* Π.

Definition 15. *Given a conditional plan P and a plefa, we define the* overlay *of P onto the plefa as the state-behaviour set graph corresponding to the possible traces of P's execution according to the environmental model specified by the plefa.*

Definition 16. *Given a conditional plan P and a problem instance <plefa, G>, we say that* P goes from I to G *iff, in the overlay of P onto the plefa, each fringe node T makes true* $satisfies(T, G)$.

Definition 17. *An rfa is* complete *w.r.t. a problem instance <plefa, G> iff* if *there exists a conditional plan P going from I to G* then *the rfa is a solution.*

Definition 18. *An rfa is* complete w.r.t. the *plefa* framework *iff, for any problem instance* Π, *the rfa is complete w.r.t.* Π.

Definition 19. *Given a problem instance <plefa, G>, a state selection function sel is* weak *iff* $\forall f \in E \;\; \forall s \in I \; (\exists r \in S(\text{path}_f(s, r) \wedge \neg\text{path}_f(r, s))) \Rightarrow s \in sel(I)$, *where* path_f *denotes that there exists an action-labeled path of length 1 in the partial state-behaviour set graph, using* $\{f\}$ *as the initial behaviour set, from state s to state r.*

Definition 20. *Given a problem instance <plefa, G>, a behaviour selection function* sel_b *is* weak *iff* $\forall s, r \in I \; (\exists f \in E(\text{path}_f(s, r) \wedge \neg\text{path}_f(r, s))) \Rightarrow f \in sel_b(F)$.

Definition 21. *Given a problem instance <plefa, G>, there is* effectory expansion *iff* $\exists f \in E \;\; \exists a \in A \;\; \exists I \subseteq S \;\; |f(a, I)| > |I|$.

Definition 22. *In a sense similar to [14] we say that the update function employs* monotonic learning *iff* $\forall F \subseteq E\ \forall I, I' \subseteq S\ \forall a \in A$ $update(F, I, a, I') \subseteq F$.

Theorem 1. *Algorithm 3 (*pl-CSEL*) is sound and complete w.r.t. the plefa framework under the following conditions:*

1. et–plan$_{pl}$ *is implemented with breadth-first search*[3]*;*
2. *the initial state set is finite;*
3. *the state selection function sel is* weak*;*
4. *the behaviour selection function sel$_b$ is* weak*;*
5. *there is no effectory expansion;*
6. *learning is monotonic.*

Proof. The theorem was proved for the pefa by Nourbahsh (see Theorems 5.2, 6.2 and 6.3 in [11]). The result can be extended to the plefa by noting that, under condition 6, the set of possible environmental behaviours eventually converges to a singleton because of the definitions of solution to a problem instance and of consistent efa. □

4 Related Work

Nourbakhsh [11], [12]) deals with the interleaving of planning and execution in much the same way as we did in Sect. 2. He introduces also abstract planning, in which a partial order of *pefa*'s is given, each corresponding to a different abstraction level. No attempt is made to learn the behaviour of the environment.

Safra and Tennenholtz ([14]) introduce a framework for planning while learning that they employ for a computational study on the subject. Their framework allows to specify a set of possible behaviours, of which only one element at a time can be selected as the possible actual behaviour of the environment. It is easy to show that their framework is a special case of our *plefa* framework.

5 Conclusions

We have presented a framework for interleaving planning-while-learning and execution and have introduced sufficiency conditions for the soundness and the completeness of the control system of an agent built upon our framework. Though supported by the encouraging results of Nourbakhsh's work, the *plefa* framework needs empirical testing in order to assess its advantages over the other approaches. Future work will be focused on the introduction of abstract planning and inductive learning in our framework.

[3] and using the Viable Plan Termination Rule ([11]).

Acknowledgments. Alessandro Provetti provided invaluable constant support throughout the whole writing of this paper. Illah Nourbakhsh gave me precious suggestions on a number of studies related to the present work and was very patient in replying to many questions about his framework. This research was partially supported by *Progetto cofinanziato (ex 40%) MURST "Agenti Intelligenti: interazione ed acquisizione di conoscenza"*.

References

[1] Marcello Balduccini and Gaetano A. Lanzarone. Autonomous semi-reactive agent design based on incremental inductive learning in logic programming. In Wiebe Van der Hoek, Yves Lesperance, and Rich Scherl, editors, *Logical Approaches to Agent Modeling and Design, Proceedings of the ESSLLI'97 Symposium*, pages 1–12, 1997.

[2] Chitta Baral and Son Cao Tran. Relating theories of actions and reactive control. *Transactions on Artificial Intelligence*, 2:211–271, 1998.

[3] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[4] Marco Dorigo. Editorial introduction to the special issue on learning autonomous robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(3):361–364, 1996. Part B.

[5] R. Fikes. Monitored execution of robot plans produces by strips. Technical Report 55, Stanford Research Institute, 1971.

[6] R. A. Kowalski. Using meta-logic to reconcile reactive with rational agents. *Meta-logic and Logic Programming*, pages 227–242, Jan 1995.

[7] John E. Laird and Paul S. Rosenbloom. Integrating execution, planning, and learning in soar for external environments. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 1022–1029, 1990.

[8] Pattie Maes. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*. MIT Press, 1990.

[9] Pattie Maes and Rodney A. Brooks. Learning to coordinate behaviours. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 796–802, 1990.

[10] David E. Moriarty, Alan C. Schultz, and John J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276, 1999.

[11] Illah Nourbakhsh. *Interleaving Planning and Execution for Autonomous Robots*. Kluwer Academic Publishers, 1997.

[12] Illah Nourbakhsh. Using abstraction to interleave planning and execution. In *Proceedings of the Third Biannual World Automaton Congress*, 1998.

[13] E. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[14] S. Safra and M. Tennenholtz. On planning while learning. *Journal of Artificial Intelligence Research*, 2:111–129, 1994.

Integrated Reinforcement and Evolutionary Learning Algorithm: Application to Alife in a Shared World

Jianjun Yan, Naoyuki Tokuda, and Juichi Miyamichi

Computer Science Department, Faculty of Engineering,
Utsunomiya University, Utsunomiya 321-8585, Japan

Abstract. The reinforcement learning algorithm we have developed for simulating Alife in a shared environment has proved to be quite effective in implementing an easy adaptation of an organism to an environment at an individual level while the evolutionary learning algorithm is effective in evolving the capacity to internally generate and develop its own teaching input to guide learning at a population level. The reinforcement signal reflecting interactions with changes in environment such as the number of food elements and their distribution facilitates the task of organisms' finding and then eating food efficiently in a shared environment at an individual level while the evolutionary process produces a useful feedback from the environment updating the teaching input. The integration of the reinforcement signal and the evolutionary process shows an excellent improvement in performance of food eating capability showing the significance of the integrated learning.

1 Introduction

Learning is perhaps one of the most fundamental processes involved in an environmental adaptation. When an organism is assumed to eat food for survival, its learning interacts both with the environment and the evolution. At an individual level, learning interacts with environment [2]. Learning can change the environment itself either by modifying the number and the distribution of food elements in the environment or by updating an organism's relative position in the environment and the environment can, in turn, guide an organism's learning through the feedback of the organism's actions, i.e. an organism can extract from the environment the necessary information on how to learn to adapt to the environment. From this point of view, learning can be considered as a continuously active process which is sensitive to environmental feedback. In this paper, we focus our attention on how environment guides learning. At a population level, on the other hand, learning interacts with evolution [1]. Learning can guide evolution by exploring approximations to the solutions sought by evolution and evolution can help learning by leaving the trace of the evolutionary change in the genotype and having the individual inherit a genome which is the cumulative result of the past evolutionary changes [2], i.e. evolution can create

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 260–271, 2000.

good environments for learning to occur(e.g. good initial weights). This paper concentrates on how evolution guides learning.

To explore how learning is guided by environment, we employ a Reinforcement Learning (RL) algorithm [3] because an organism is able to use environmental cues as reinforcement signals. The reinforcement signal can be used to determine a contradicting information directly from the association between inputs and actions in the environment. Meanwhile, the reinforcement signals in our RL develop a useful strategy for the organisms to learn to compete fairly for food in the shared environment.

The evolutionary learning(EL) approach [1,2] is used to facilitate organisms to develop their own internal teaching input through evolution so that their corresponding neural networks generate the internal teaching input by transforming the network inputs through connection weights that are evolved using genetic algorithm.

In a real world, learning may interact both with the environment and the evolution simultaneously at different levels. It is most instructive to integrate the RL algorithm with the EL approach, in which, the evolutionary approach is applied to generate and develop the internal teaching input at a population level and RL is employed to further update the internal teaching input by interacting with the environment at an individual level. The integrated learning shows an improved performance of eating food in its corresponding simulation.

Although we address an artificial life application in this paper, the proposed integrated algorithm is general enough to be applicable to many other applications which can be implemented by neural network learning algorithm.

2 Learning Task

We let a population of organisms strive to survive by finding food in a simulated environment as discussed by Nolfi, Elman and Parisi [1]. The shared world comprises 100×100 cells and initially contains 1,000 randomly distributed food elements with each occupying a single cell. The population consists of 100 organisms and each organism is allowed to live for a total of 6,500 actions in the environment.

At any particular moment, any of the 100 organisms occupies one of the cells. The organism has a facing direction and a rudimentary sensory system sensing the angle and distance to the nearest food element as input from the environment. The organism may, in a single action, turn either 0 degrees, 90 degrees right, 90 degrees left, or 180 degrees and then go one cell forward in the environment. When an organism happens to step on a food cell, it eats the food element and this element will disappear. All the food elements will be periodically reintroduced each average of 100 actions of the organisms.

Each organism's nervous system is modeled by a fuzzy neural network(FNN) comprising three layers of neurons. The FNN has 2 input variables labeled as X_0, X_1 which give the angle(measured clockwise from the organism's facing direction) and the Euclidean distance to the nearest food element (both values are

scaled from 0.0 to 1.0), and 2 output variables labeled as Y_0, Y_1 which are encoded in a binary fashion: 00=go one cell forward, 10=turn 90 degrees right and then go one cell forward, 01=turn 90 degrees left and then go one cell forward, 11=turn 180 degrees and then go one cell forward. Each output value produced by the FNN is thresholded to either 1 or 0. In order to use fuzzy logic, we must divide the domain of every input variable into some fuzzy intervals. In general, we initially suppose there are four fuzzy intervals $\{East, South, West, North\}$ on X_0, labeled as $\{X_{0,0}, X_{0,1}, X_{0,2}, X_{0,3}\}$ and two fuzzy intervals $\{Far, Near\}$ on X_1, labeled as $\{X_{1,0}, X_{1,1}\}$. Figure 1 shows the architecture of the FNN.

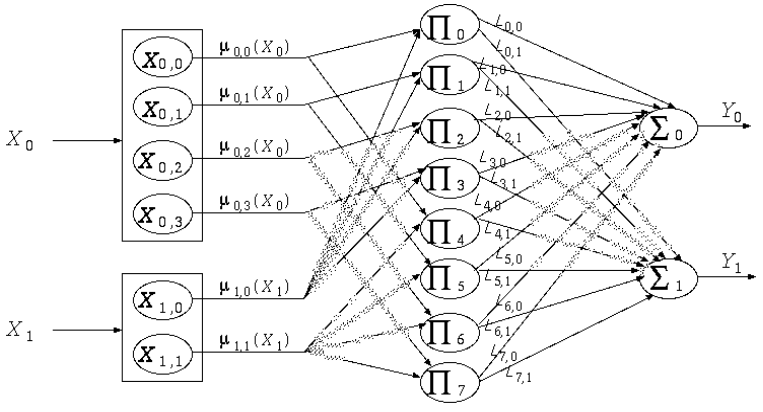


Fig. 1. The architecture of FNN

Here $\prod_i$(i=0,1,2,...,7) is the multiplication neuron, $\sum_i$(i=0,1) is the summation neuron, while $\mu_{0,j}(X_0)$(j=0,1,2,3) and $\mu_{1,j}(X_1)$ (j=0,1) represent the respective degree of membership function for X_0 and X_1 for the given fuzzy intervals. $L_{i,j}$ (i=0,1,2,...,7; j=0,1) is the connection weight between the second layer and third layer. Every first layer neuron outputs the membership degree $\mu_{0,j}(X_0)(j = 0, 1, 2, 3)$ and $\mu_{1,j}(X_1)(j = 0, 1)$. For example, $\mu_{0,0}(X_0)$ is the membership degree to which X_0 belongs to $X_{0,0}$. The inferences of second layer neurons can be made clear by interpreting them, say, neuron $\prod_1$ as a typical fuzzy inference rule in terms of *if-then-rule:*

IF X_0 *is* $X_{0,1}$ *AND* X_1 *is* $X_{1,0}$ *THEN*
output of $Y_0(X_0, X_1)$ *is* $L_{1,0}$ *AND output of* $Y_1(X_0, X_1)$ *is* $L_{1,1}$

When input X_0,X_1 are given, the neurons in the third layer output

$$Y_j(X_0, X_1) = \sum_{i=0}^{7} (\mu_{0,i(mod\,4)}(X_0) \times \mu_{1,i(mod\,2)}(X_1)) \times L_{i,j}, \quad j = 0, 1 \qquad (1)$$

The learning task is to make the organisms find and then eat food efficiently in the environment. The number of food elements eaten by an organism during its lifetime comprises its fitness value used in the selection process.

3 Learning from Environment with Reinforcement Signal

Learning interacts with environment and can be considered as a continuously active process which is sensitive to environmental feedback[3]. Because an organism is able to use environmental cues as reinforcement signals, we employ a Reinforcement Learning (RL) paradigm to explore how learning is guided by environment. In our simulated environment, an increase in the number of food elements eaten corresponds to a reward, otherwise to a penalty. Thus, we use a simplified version of associative reward-penalty algorithm[3] as the implemented RL algorithm, in which a food eaten criterion is employed to generate a two-valued reinforcement signal.

3.1 Reinforcement Learning(RL) Algorithm

Figure 2 depicts the reinforcement learning(RL) model. In our simulated envi-

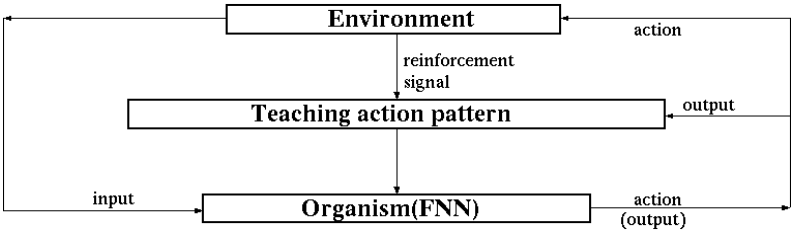


Fig. 2. The reinforcement learning(RL) model

ronment, we expect the organisms to be able to eat as much food as possible. If an ideal result is achieved, the organisms can eat the whole food elements available in their entire lifetime of total actions, i.e. each organism can eat one food element by each $\Delta\alpha$ actions, on average.

$$\Delta\alpha = \left[\frac{\text{the number of total actions during life}}{\text{the number of total food elements available during life}}\right]$$
$$= \left[\frac{6500/_{\text{organism}} \times 100\,\text{organisms}}{1000/_{100\,\text{actions}} \times 6500\,\text{actions}}\right] = \left[\frac{6500 \times 100}{1000/_{100} \times 6500}\right] = 10$$

Thus, we can describe our reinforcement learning(RL) algorithm for each organism(FNN) as below.

At the end of each $\Delta\alpha$ actions, a two-valued reinforcement signal S is generated:

S=+1, if the organism eats at least one food element within $\Delta\alpha$ actions;

S=-1, otherwise.

This is used to produce a teaching action pattern $Y_i^d(i = 0, 1)$ from the last output $Y_i(i = 0, 1)$ of the FNN:

$$Y_i^d = \begin{cases} H(Y_i - \frac{1}{2}) & S = +1 \\ 1 - H(Y_i - \frac{1}{2}) & S = -1 \end{cases} \quad (2)$$

where H is the Heaviside step function and the output Y_i is explained by the equation (1) and has not yet been thresholded to 0 or 1. The Steepest Descent Learning Method(SDLM) [5] is then used to update the weights of the FNN on the basis of a discrepancy between the teaching action pattern $Y_i^d (i = 0, 1)$ and the output $Y_i (i = 0, 1)$ of FNN, where the discrepancy is defined as $E = \frac{1}{2} \sum_{i=0}^{1} (Y_i^d - Y_i)^2$.

At the end of life, the neural networks are evolved by the genetic algorithm described in the Sect. 5.1 but here we adopt the network architecture of Fig. 1. This process continues for a given number of generations (say, 200 generations).

4 Evolutionary Learning(EL) with Teaching Input

One way of having the organisms improve their learning performance (i.e. develop a better food eating ability) would be to use supervised learning algorithms such as back-propagation algorithms. But a requirement for the learning mechanism in our shared environment is the assumption that there be no external teaching input available to the organisms. This would seem to preclude many successful back-propagation algorithms. However, the organisms can develop their own internal teaching input through the evolution process. By having a population of organisms to be reproduced selectively on the basis of their ability to find food and by adding mutations in order to maintain a sufficient basis of variability in the population, we should see an efficient food approaching behavior emerge across generations[1]. Our strategy of developing the teaching input is based on the above concept inspired by Nolfi and Parisi[1]. However, instead of living in an isolated environment, the organisms (represented by FNNs) inhabit in a shared environment in our simulation.

4.1 Individual Learning from Teaching Input

If the FNN(shown in Fig. 1) is able to learn from internal teaching input, it must necessarily be capable of modifying its architecture by providing additional output neurons for encoding the internal teaching input. This can be done in many different ways and the simplest one is to provide an additional set of hidden neurons(i.e. the second layer neurons). The additional set of hidden neurons receive connections from the first layer neurons and send connections to the teaching input neurons. This is called "teaching sub-network" and the remainder "standard sub-network"[1]. Figure 3 shows the architecture of the FNN with internal teaching input, where the FNN is made up of two sub-networks, a standard sub-network and a teaching sub-network. The two sub-networks have the same architecture which is described in Fig. 1 and they share the first layer neurons but each has a separate set of eight hidden neurons and two output neurons with randomly initiating weights.

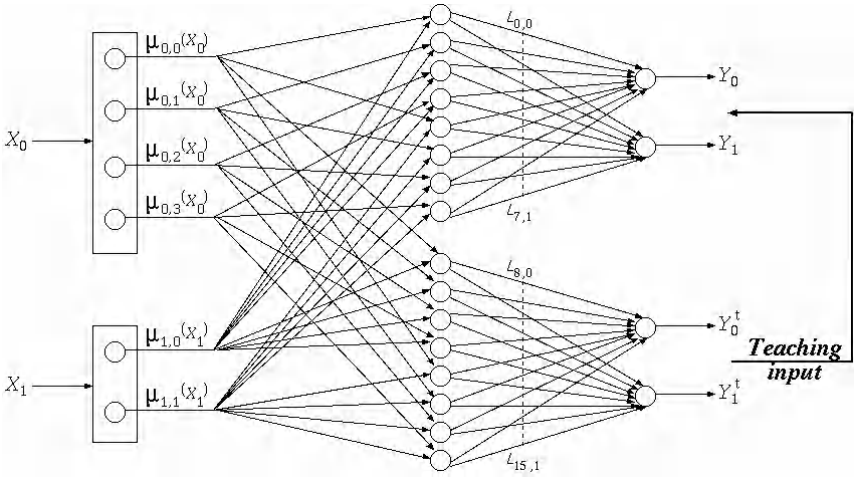


Fig. 3. Architecture of the FNN with internal teaching input

The output $Y_j (j = 0, 1)$ of the standard sub-network, computed by the equation (1), is used to move the organism in the simulated environment, while the output $Y_j^t (j = 0, 1)$ of the teaching sub-network, calculated also by the equation (1) with the weights varying from those used in computing Y_j, is employed as teaching input to the corresponding output of the standard sub-network.

As soon as Y_j and $Y_j^t (j = 0, 1)$ have been computed, the network is ready to modify the connection weights of the standard sub-network through the SDLM [5] on the basis of the discrepancy between Y_j and $Y_j^t (j = 0, 1)$. The connection weights of the teaching network will remain the same.

4.2 Population Evolution by Genetic Algorithm

Chromosome Encoding. Conventional genetic algorithms use a discrete chromosome coding method such as binary coding, representing the discrete values in the parameter space. They are difficult to adjust to the parameters strictly according to the distribution of the data. In order to overcome this difficulty, we directly code the chromosome with a set of real float values which are the weights of the FNN. Figure 4 shows the chromosome architecture.

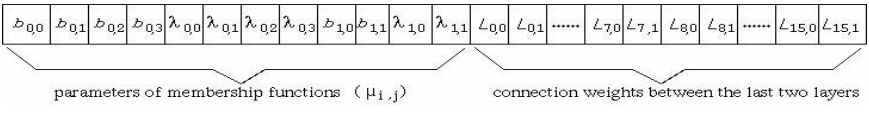


Fig. 4. Chromosome encoding

Population Initiating. We begin with a population of 100 organisms each having the architecture of Fig. 3 with random weights, selected between -1.0 and 1.0, on all its connections. Each of the 100 organisms is allowed to live for a total of 6,500 actions in the shared environment. At the end of lifecycle, the fitness value of each organism is calculated based on the total number of food elements eaten by the organism during life.

Selection. The 100 individuals are ranked in terms of their fitness and the best 50%, i.e. the best 50 organisms are selected to do crossover operator.

Unfair Average Crossover. Traditional genetic algorithms usually use one or multi-point crossover with bit strings incurring an extremely high costs for the crossover. Although some genetic algorithms use numerical coding and an average operator in crossover operation [6], this does not maintain a badly needed variability in property among the population. We have decided to apply the

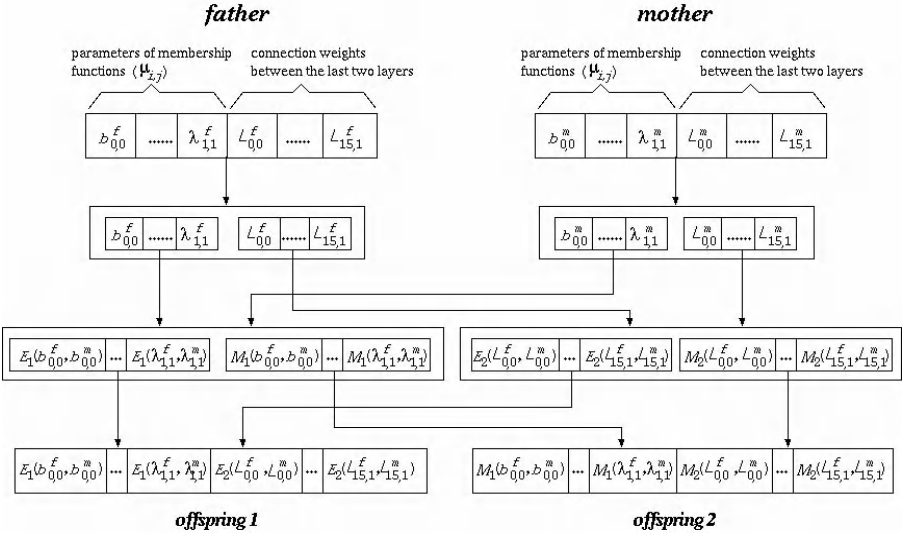


Fig. 5. Unfair average crossover(UFAC)

Unfair Average Crossover(UFAC) of Nomura and Miyoshi [4], into the evolution process because of its higher capacity of adaptation for environment. UFAC is applicable to real float value chromosomes, which allows a higher capacity of adaptation for environment than other conventional methods reproducing offsprings far from the parents. Figure 5 shows the UFAC process where two offsprings are made from two selected parents. We cut each chromosome of the two selected parents into two parts, one of which only includes the parameters of

the membership functions and the other contains the connection weights between the last two layers in the FNN. In Fig. 5, E_1, E_2, M_1, M_2 are defined as follows.

$$E_1(x_1, x_2) = (1 + \frac{1}{2})x_1 - \frac{1}{2}x_2, \qquad E_2(x_1, x_2) = -\frac{1}{2}x_1 + (1 + \frac{1}{2})x_2$$
$$M_1(x_1, x_2) = (1 - \frac{1}{2})x_1 + \frac{1}{2}x_2, \qquad M_2(x_1, x_2) = \frac{1}{2}x_1 + (1 - \frac{1}{2})x_2 \quad (3)$$

Here $E_1(x_1, x_2)$ gives a value much different from an average of the parents in the direction of the father's value(x_1), while $E_2(x_1, x_2)$ giving far from the average in the direction of the mother's value(x_2); $M_1(x_1, x_2)$ and $M_2(x_1, x_2)$ are equal to the average of the parents. Thus in UFAC, if the parents have some appropriate characters(in terms of learning skills, say), one of the offsprings (*offspring 2*) inherits them on the average. In the meantime, a variety in the population is maintained by producing offspring(*offspring 1*) far from the average.

Mutation. Each of the best 50 organisms selected by selection operator and the new 50 organisms generated by the crossover operator will have mutation by selecting 10% of genes in its chromosome at random and then perturbing them by adding a random real float value between -1.0 and +1.0.

Replacement. The worst 50 organisms with the smallest fitness values in the current population are replaced with the 50 newly generated organisms by the crossover operator. This forms the next generation of new 100 individuals.

Stopping Condition. The evolutionary process continues for a given number of generations(say, 200 generations).

5 Integrated Learning(IL) Algorithm with Reinforcement Signals Updating Teaching Input

In this section, we try to integrate the RL with the EL so as to make full use of their respective advantageous characteristics, where RL is exploited to learn the teaching sub-network enabling the teaching input to be updated in time during life with an EL initiated at the end of life in force while the teaching input will be then used to guide the organism's actions during life. Figure 6 shows the integrated learning(IL) model, where the standard sub-network and teaching sub-network assume an identical form as illustrated in Fig. 3.

5.1 The Algorithm

The integrated learning(IL) algorithm is given below.

- **Step 1.** Initiate a population of 100 organisms' neural networks, each having the IL architecture of Fig. 6 and a randomly assigned set of connection weights all selected between -1.0 and 1.0. Each organism will then respectively take the following steps of Step 2 to Step 5.

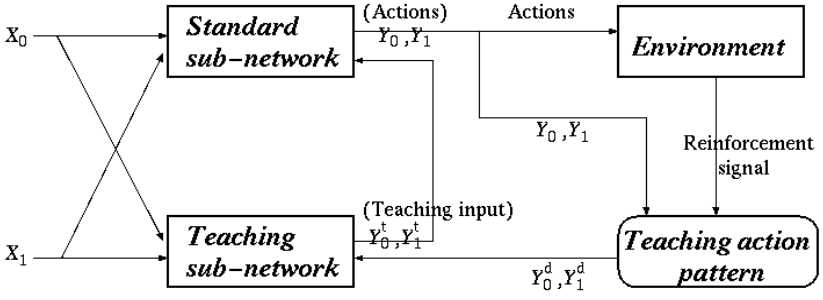


Fig. 6. The integrated learning(IL) model

- **Step 2.** Receive sensory input X_0, X_1 and compute the output $Y_j(j = 0, 1)$ of the standard sub-network and the output $Y_j^t(j = 0, 1)$ of the teaching sub-network by the equation (1) with their corresponding weights respectively.
- **Step3.** Calculate the discrepancy between $Y_j(j = 0, 1)$ and $Y_j^t(j = 0, 1)$ by $E = \frac{1}{2}\sum_{j=0}^{1}(Y_j^t - Y_j)^2$ and use it to modify the weights of the standard sub-network by the SDLM [5].
- **Step 4.** Perform the action designated by the output $Y_j(j = 0, 1)$ of the standard sub-network.
- **Step 5.** At the end of each $\Delta\alpha$ actions($\Delta\alpha$ being selected as in Sect. 3.1), generate a two-valued reinforcement signal S:

  S=+1, if the organism eats at least one food element within $\Delta\alpha$ actions
  S=-1, otherwise

  Signal S is used to produce a teaching action pattern $Y_i^d(i = 0, 1)$ from the last output $Y_i(i = 0, 1)$ of the standard sub-network by the equation (2). The teaching sub-network is then ready to update its connection weights by the SDLM [5] on the basis of the discrepancy E. $E = \frac{1}{2}\sum_{i=0}^{1}(Y_i^d - Y_i^t)^2$.
- **Step 6.** At the end of life, evolve the neural networks by the genetic algorithm stated below.
- **Step 7.** Go to step 2.
- **Stopping condition:** The above process continues for a given number of generations(say, 200 generations)
- **Genetic algorithm:** The chromosome encoding method is the same as that in Sect. 4.2. The population comprises 100 organisms and each organism is allowed to live for 6,500 actions in the shared environment. At the end of life, the 100 individuals are ranked in terms of their fitness(total number of food elements eaten during life) and the best 50%, i.e. the best 50 organisms are selected to do Unfair Average Crossover as described in Sect. 4.2. Each of the best 50 individuals and the 50 individuals generated by the crossover operator will have mutation by selecting 10% of genes in its chromosome at random and then perturbing them by adding a random real float value between -1.0 and +1.0. At last, the worst 50 organisms with the smallest fitness values in the current population are replaced with the 50 newly generated organisms.

6 Simulation Results

Figure 7 shows the average number of food elements eaten by the total 100 organisms. All the graphs in this section represent the average results of 20 simulations of the same procedure but starting with different randomly assigned weights at generation 0. These results are compared with those we obtain without learning in the simulation. Figure 7 indicates that all the three learning curves attain much higher level of average food eaten than those without learning. Undoubtedly these learning methods have been effective in significantly improving the eating performance of organisms.

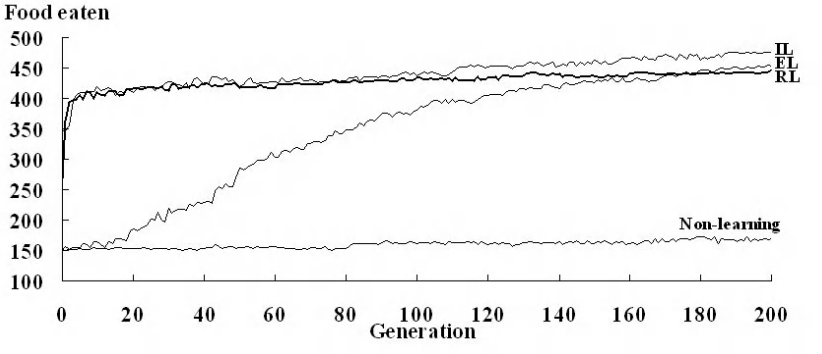


Fig. 7. Average food eaten throughout generations

We see that IL outperforms RL, implying the positive role of the evolutionary procedure in IL. On the other hand, Fig. 7 shows that IL has improved the eating performance of the EL. This improvement comes from RL used in IL. Thus in IL, both RL and EL procedures have beneficial effect on the ability of eating food. RL produces a useful feedback to learn the teaching sub-network and the evolutionary procedure develops the teaching input to yield an efficient guide to the organism's actions. The evolutionary phase in essence generates a rough guide effective at the population level to the organisms, while the RL phase in providing fine feedback from the environment at the individual level so that the organism learn in detail.

From Fig. 7, we can also see that RL produces a better average curve in the first 170th generations while the EL outperforms the RL in later generations. It shows, on one hand, the effect of RL is immediate in improving food eating capability but suffers easily from the stagnating performance in the later generations because the reinforcement signal is difficult to provide enough feedback information to further improve the performance of eating food. On the other hand, the effect of EL is felt too slow, catching up and surpass the RL much at later generations because the organisms can develop more useful teaching input in proportion to the length of successive generations while the evolution takes place.

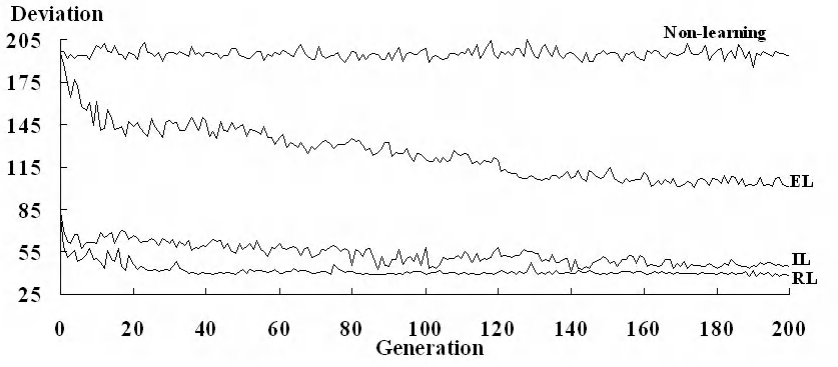


Fig. 8. The standard deviation of the food eaten throughout generations

Figure 8 shows the standard deviation of food eaten by the 100 organisms from their average food eaten. Standard deviation is apparently least for the population guided by RL population. RL provides an extremely effective strategy for minimizing deviations among organisms implying that a very fair competition is being ensured among the organisms. This can be explained by its goal-directed learning strategy adopted. Without making any use of the nearness information of the food element which naturally gives more efficient information for our purpose in improving an individual's fitness values, RL does evaluate the effectiveness of the last action taken and enforce the neural networks to be trained constantly increasing the possibility of eating food element at a next action blindfoldedly so to speak. It implies that a failure of one organism at one time leads to its success at a next or later time step but only by sacrificing the chances of other organisms.

Unlike RL, figure 8 also shows that there is a larger deviation for the EL population. The higher deviation of EL curve is due to the lack of a fair competition strategy just as in the case without learning. The lower level of the IL comes from the involved RL which has provided a fair competition strategy to reduce the individual difference in the population. We see the level of IL curve is higher than that of RL curve. In the IL, the involved RL produces a teaching action pattern for the teaching sub-network to learn and the teaching sub-network is developed to guide the organism's actions. Thus, the involved RL in the IL only provides an indirect guide for the organism's action through the teaching sub-network. The reinforcement signal is then delayed in such an indirect guidance and it increases the individual difference in the population.

7 Conclusion and Future Work

Taking advantage of both the RL algorithm and the EL approach, the integrated learning (IL) algorithm, has achieved the best food eating ability in the simulations. It shows both the adaptation of the evolution and the feedback of the

environment contribute to the improvement of the eating performance, where the environment sends back a reinforcement signal in time to learn the teaching sub-network and the evolution develops the teaching input to yield an efficient guide to the organism's actions. The evolutionary phase can be regarded as a process of producing a rough guide at the population level to the organisms, while the RL phase is considered as a process of providing fine feedback from the environment at an individual level.

The current simulation is based on the neural network model. Often the learning performance suffers from the stagnated saturation especially in the later generations due to the limited capability of the fixed architecture of the neural networks. The more flexible constructive algorithm allowing a dynamic network architecture should improve this stagnant behavior and we are now investigating the possibility.

References

1. S. Nolfi and D. Parisi, "Auto-teaching: networks that develop their own teaching input", Proceedings of the Second European Conference on Artificial Life, eds., J. Deneubourg, H. Bersini, S. Goss, G. Nicolis and R. Dagonnier, Brussels, 1993.
2. S. Nolfi and D. Parisi, "Learning to adapt to changing environments in evolving neural networks", Adaptive Behavior, Vol.5, No.1, pp. 75-98, 1997.
3. A. Barto and P. Anandan, "Pattern recognizing stochastic learning automata", IEEE Transactions on Systems, Man, and Cybernetics, Vol.15, pp.360-375, 1985.
4. T. Nomura and T. Miyoshi, "Numerical coding and unfair average crossover in GA for fuzzy rule extraction in dynamic environment", Fuzzy Logic, Neural Networks and Evolutionary Computation, eds., T. Furuhashi and Y. Uchikawa, Springer-Verlag, pp. 55-72, 1996.
5. J. Yan, N. Tokuda and J. Miyamichi, "A new constructive compound neural networks using fuzzy logic and genetic algorithm: 1 application to artificial life", IEICE TRANS. INF. & SYST., Vol. E81-D, No.12, pp. 331-340, December 1998.
6. L. Davis, "Handbook of genetic algorithms", Van Nostrand Reinhold, 1990.

RuLess: A Method for the Acquisition and Simplification of Rules

Pablo R. de Buen Rodriguez[1], Eduardo F. Morales[2], and Sunil Vadera[3]

[1] Instituto de Investigaciones Eléctricas, Av. Paseo de la Reforma 113, Temixco, Mor., 62490, Mexico; Tel. (7)318-38-11 ext.7797; Fax: (7)318-25-38;
debuen@iie.org.mx
[2] ITESM - Campus Morelos, Av. Paseo de la Reforma 182-A, Temixco, Mor., 62589, Mexico;
emorales@campus.mor.itesm.mx
[3] Department of Computer and Mathematical Sciences, University of Salford, Salford, M5 4WT, UK;
S.Vadera@cms.salford.ac.uk

Abstract. An algorithm developed to help an expert generate rules is presented. The algorithm, which has been called RuLess, consists of two main stages: (i) a session to incrementally capture the rules, and (ii) a mechanism to simplify the rules. In general, it can be used on environments where there is a finite set of possible examples which are not available in advance, and from which a set of classification rules needs to be produced. It is useful in domains in which all the attributes are discrete and the number of examples is not too large, as the user needs to manually classify all the examples. RuLess was used to generate the tutoring rules of LacePro, a multifunctional system to learn, apply and consult established procedures. Based on the rules obtained for LacePro, the RuLess method was compared against the CN2 and Ripple Down Rules methods, which are two well-known rule generation procedures.

1 Introduction

One main step in the development of a knowledge-based system is the knowledge acquisition process. This process has the difficulty that, in general, experts can not provide or articulate all the necessary knowledge in advance. Hence, several techniques have been proposed in the literature to ease this activity [4]. This paper presents RuLess , an algorithm developed to help an expert generate the rules used to define the tutoring agenda of LacePro [2], a multi-functional system to to learn, apply and consult established procedures [1]. The algorithm consists of two main stages: (i) a session to incrementally capture the rules, and (ii) a mechanism to simplify the rules.

The paper is organised as follows. Section 2 contains a detailed description of RuLess, as implemented in LacePro. Section 3 presents an experiment to

[1] This work was developed under the support of CONACYT (Mexican government's council for science and technology).

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 272–283, 2000.

evaluate the performance of the algorithm. The results obtained are compared with those generated by the CN2 method [1].

Finally, Section 4 gives some conclusions about RuLess and contrasts this algorithm with CN2 and the Ripple Down Rules method [3] which, as CN2, is a well-known knowledge acquisition methodologie that share some characteristics with RuLess.

2 The RuLess Algorithm

Figure 1 shows the architecure of LacePro. By using LacePro, professional engineers can learn, apply and consult established procedures. Actually, the architecture enables them to combine these three tasks in order to satisfy her own requirements under a given situation. For instance, depending on how familiar she is with a procedure, a user can start applying the procedure to solve a problem, and then move to the consulting mode to inspect specific information, or move to the tutoring mode in order to have a full tutoring session about that procedure. When learning a procedure, she can receive a full tailored session that includes explanation of the steps and theoretical concepts, examples and evaluations; or she can learn the procedure by solving problems or by navigating by herself through the domain knowledge; or by a combination of these two methods.

Under the above context, one of LacePro's key aims is to adapt to a user needs and experience. One of the ways it achieves this adaptibility is to propose a specific agenda for learning or using a procedure. To define this agenda, LacePro uses a set of rules generated by an expert, which are based on the general characteristics of procedures and on the general characteristics of possible users. Given that the number of cases that result from the combination of these characteristics is considerable, it was decided to provide the expert with a mechanism to help her to generate those rules. RuLess, which is the mechanism developed for this task, can, in general, be used on environments where there is a finite set of possible examples (i.e., cases) which are not available in advance and from which a set of classification rules need to be produced. It is useful in domains in which all the attributes are discrete and the number of examples is not too large, as the user needs to classify all the examples. Its main characteristics are the following: (a) the cases and the rules are generated in paralel within a single process, (b) during the process, the rules that have already been defined are used to pre-classify the new generated cases, which quickens the work of the expert, and (c) the resulting rules reflect the expert criteria and are simple and comprehensible, this facilitates the maintenance of the rules. The first two or these characteristics result in a relatively fast generation of rules.

RuLess receives as input, the possible characteristics of the users and of the procedures in the form of attributes names (e.g., importance of the procedure, type of user, etc.), and the possible attribute values (discrete values like: high, intermediate). The resulting rules are of the form 'if <conjunctive conditions> then $< Agenda_n >$', where each condition may have one or more disjunctive

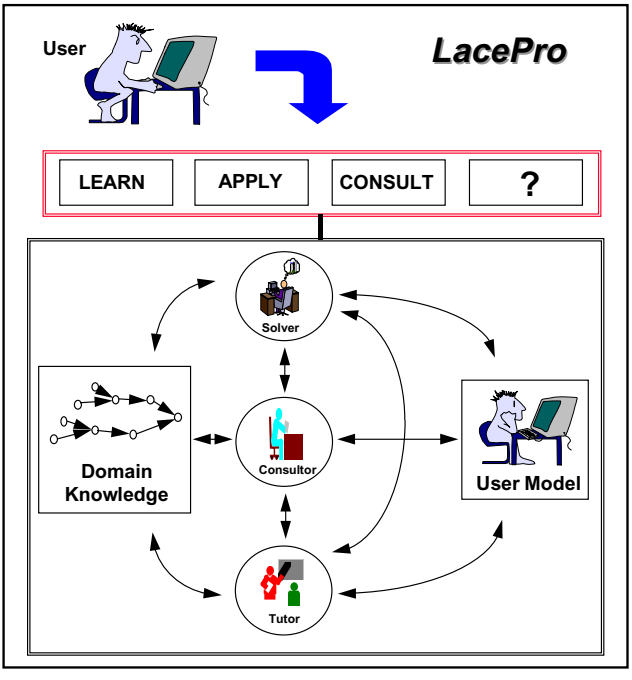


Fig. 1. Architecture of LacePro.

values (i.e., a condition is a combination of an attribute and one or more of its values forming an affirmative statement; e.g., the size of the procedure is big or small) and $< Agenda_n >$ represents a possible tutoring agenda. The algorithm proceeds in two stages. First, examples are generated and an initial set of rules is learned. A second stage simplifies the rules. The first stage proceeds as follows.

1. A new example is generated by assigning values to the attributes. The values are chosen following a prefixed order and cycling through the alternative values to generate new examples. This in such a way that, at the end of the whole process, all the possible combinations of attributes and values are considered.
2. The current set of rules is used to classify the new example. The rules are attempted, one by one, from the first towards the last, until either a rule succeeds, or no rule is applicable.
3. If there is no applicable rule, or the expert disagrees with the classification, she is invited to add a new rule. RuLess checks that the new rule does not adversely affect the classification of previously classified examples and warns the expert if it does. The rules are ordered so that more specific rules (i.e., the rules with more conditions) are placed earlier than more general rules.

Once these initial rules are obtained, the second stage performs a post-processing operation to simplify the rules. This involves eliminating unnecessary

rules, conditions and values of attributes. This post-processing consists of the following steps.

1. Unfold rules that contain disjunctive values. For all the rules that contain attributes with more than one value, generate a rule for each possible combination of values. The result is a set of rules with only one value in each of its conditions. For instance, unfolding the rule:

```
Rule 1: if size is big or normal
        and goal is learn or apply
        then Agenda1
```

   will produce the rules:

```
Rule 1a: if size is big
         and goal is learn
         then Agenda1

Rule 1b: if size is normal
         and goal is learn
         then Agenda1

Rule 1c: if size is big
         and goal is apply
         then Agenda1

Rule 1d: if size is normal
         and goal is apply
         then Agenda1
```

   This unfolding facilitates the elimination of unnecessary values (expressed as single conditions) through the second step explained below.
2. Eliminate unnecessary conditions and rules. For all the rules, and for all the conditions of the rules, the influence of each condition is evaluated. If when eliminating a given condition, there is no change in the classification of any of the examples, that condition is not required, thus it is eliminated. A rule is eliminated when all its conditions are redundant (i.e., the rule is never used). This means that the examples covered by that rule are already covered by the rules that are above it in the list, and thus it can be eliminated.
3. Try to join similar rules that have almost the same conditions but differ on one value of an attribute (this is the inverse situation of the first step). For instance, as shown below, *Rule 1* and *Rule 2* can be merged to produce a new rule.

Rule 1:	Rule 2:	New Rule:
if size is big	if size is big	if size is big
and goal is learn	and goal is apply	and goal is learn or apply
and user is novice	and user is novice	and user is novice
then Agenda1	then Agenda1	then Agenda1

In addition to the above steps, and in order to make the set of rules more comprehensible and maintainable, a process to locate and make explicit the direct relations between rules is performed. In this process, the conditions of each rule are compared with the conditions of those rules that are below it. If the conditions of this rule include all the conditions of a rule below, the former

RULE No. 27 (Specific case of RULE No. 40)
If the user is Experienced
and the goal of the user is Learn
and the importance of procedure is Low
and the average difficulty of steps is Low
and the size of procedure is Normal or Small
then A4: Brief, Introduction, Examples and Review

RULE No. 40
If the goal of the user is Learn
and the importance of procedure is Low
and the average difficulty of steps is Low
then A2: Brief Introduction, Concepts & Steps, Examples, Restrictions and Test

Fig. 2. Example of related rules.

is a specific case of the latter, and that is indicated by a note attached to the former of these rules. As an example, two related rules are shown in Fig. 2.

The above steps ensure that the simplified rules are consistent with the original rules with respect to the training examples. The extent to which they result in simplification will be addressed empirically in Section 3. The two simplifying steps of this algorithm (i.e., the second and the third steps) are now stated more formally and in greater detail using bit-wise operations. The notation used is:
$\mathbf{C}$ = classes (i.e., possible classifications),
$\mathbf{R}$ = rules,
NC = number of classes,
NR = number of rules,
NE = number of examples,
$R_{i,k}$ = i–th. rule of class k.

The algorithm for eliminating redundant conditions is given in Table 1 and the one for joining rules is given in Table 2.

As shown in Table 1, in order to eliminate the surplus conditions of the rules, for all the classes ($\mathbf{C}$) the mechanism creates bit sets (BSC_k) of a length equal to the number of examples (NE) with 1's if the example is of the particular class (k) and 0's otherwise. It also creates bit sets ($BSR_{i,k}$) of length NE for all the examples covered by each rule, with 1's in the covered examples and 0's otherwise. For each condition of each rule, the new examples covered by the rule without the condition are obtained. If in the new covered examples, there are examples of other classes, the mechanism checks if those examples are covered by the rules of those classes which are before the new rule (remember that the rules are ordered). If that is the case or if the new rule covers only additional examples of the same class, then it is safe to eliminate that condition.

Table 1. Algorithm for eliminating redundant conditions.

Given:
$BSR_{n,k}$ = all the bit sets of the rules
BSC_k = all the bit sets of the classes
For all $R_{i,k} \in \mathbf{R}$
For all the conditions (c_n) of rule $R_{i,k}$
Let $R'_{i,k} = R_{i,k} - \{c_n\}$
Let $BSR'_{i,k}$ = bit set of $R'_{i,k}$
% Isolate new covered examples
Let $NewC$ = bitwise XOR$(BSR'_{i,k}, BSR_{i,k})$
% If the new covered examples are only of class k, eliminate the condition
If $NewC$ = bitwise AND$(NewC, BSC_k)$
Then set $R_{i,k} = R'_{i,k}$
Else For each class C_l, where $l \neq k$
% Extract from the new covered examples those of other classes
Let $NewOC_l$ = bitwise AND$(NewC, BSC_l)$
If $NewOC_l \neq 0$
% Collect all the examples covered by rules of class l located before $R_{i,k}$
Then Let $Btmp = 0$
For each $R_{j,l} \in \mathbf{R}$ for class $l \neq k$ and $j < i$
$Btmp$ = bitwise OR$(Btmp, BSR_{j,l})$
% If they include the examples covered by new rule, eliminate condition
If $NewOC_l$ = bitwise AND$(NewOC_l, Btmp)$
Then $setR_{i,k} = R'_{i,k}$

The algorithm to join similar rules can be explained as follows. Suppose there are two rules $R_{i,k}$ and $R_{j,k}$, that are of the same class k (i.e., have the same conclusion), are similar (i.e., are of the same length and differ on at most one value of an attribute), and $i < j$. The aim is to replace these rules by a new, merged rule, $R_{m,k}$. But since the order of the rules is significant, the new rule needs to be placed carefully so that the coverage of the rules remains equivalent to that of the original rules. RuLess first tries to put $R_{m,k}$ in the place of $R_{j,k}$. In this case, it needs to check that the rules that are between $R_{i,k}$ and $R_{j,k}$ in the rule list do not cover any of the examples covered by $R_{i,k}$. This is because placing $R_{m,k}$ in the place of $R_{j,k}$ is equivalent to moving $R_{i,k}$ down to $R_{j,k}$. Otherwise, RuLess tries to put $R_{m,k}$ in the place of $R_{i,k}$. In this case, the system needs to check if $R_{j,k}$ does not cover examples of other classes that are covered by the intermediate rules. In either case, $R_{i,k}$ and $R_{j,k}$ are eliminated from the rule list and replaced by $R_{m,k}$.The process continues until no more possible simplifications can be made.

RuLess follows a hill-climbing approach and does not guarantee to find, from a set of original rules, the optimal set of reduced rules. It assumes a correct classification by the user in a noisless environment.

Table 2. Algorithm for joining rules.

Given:
  $BSR_{n,k}$ = all the bit sets of the rules
  BSC_k = all the bit sets of the classes
For all rules $\in \mathbf{R}$
  Let RL_k = set of rules of the *same length* and of the *same class*
  Let PR = pairs of rules, $R_{i,k} \in RL_k$ and $R_{j,k} \in RL_k$ that
    differ on at most one value of an attribute (let $a = v_1$ and
    $a = v_2$ be the different conditions)
% Suppose that $R_{i,k}$ is before $R_{j,k}$ in the rule list
% Collect all the examples covered by the intermediate rules
  Let $Btmp = 0$
  For each $R_{n,l} \in \mathbf{R}$ for class $l \neq k$ and $i < n < j$
    $Btmp$ = bitwise OR$(Btmp, BSR_{n,l})$
% Extract the examples of class k covered by such rules
  Let $BSOC_k$ = bitwise AND$(Btmp, BSC_k)$
% Extract only the examples of class k covered by $R_{i,k}$
  Let $BSR'_{i,k}$ = bitwise AND$(BSR_{i,k}, BSC_k)$
% If they do not cover an example of class k covered by $R_{i,k}$
% Then it is safe to join the rules and put the new rule in the place of $R_{j,k}$
  If bitwise AND$(BSOC_k, BSR'_{i,k}) = 0$
  Then call *join_rules*$(R_{j,k}, R_{i,k}, a, v_1, v_2)$
% Else check if the examples covered by the lower rule do not
% include examples of the intermediate rules
% First extract examples of other classes covered by the lower rule
  Else Let $BSRO_{j,k}$ = bitwise AND ($BSR_{j,k}$, bitwise NOT (BSC_k))
    If $BSRO_{j,k} = 0$ Then call *join_rules*$(R_{i,k}, R_{j,k}, a, v_1, v_2)$
% Select the covered examples of other classes of the intermediate rules
    Else Let $Btmp'$ = bitwise AND ($Btmp$, bitwise NOT (BSC_k))
% If the examples covered by the lower rule are not covered by rules
% of class $l \neq k$ that are between $R_{i,k}$ and $R_{j,k}$ then join them
% and put the new rule in the place of $R_{i,k}$
      If bitwise AND ($BSRO_{j,k}, Btmp'$) $= 0$
      Then call *join_rules*$(R_{i,k}, R_{j,k}, a, v_1, v_2)$

Table 3. Procedure to join two rules.

Procedure *join_rules*$(R_{i,k}, R_{j,k}, a, v_1, v_2)$
% Puts the new rule in the place of the rule of the first argument and
% eliminates the other rule
  Let $R_{i,k} = R_{i,k} - \{a = v_1\} \cup \{a = v_1 \vee v_2\}$
  Set $\mathbf{R} = \mathbf{R} - R_{j,k}$

If when applying the simplifying algorithms no redundant conditions in the rules are found, or it is not possible to join rules, the resulting rules are equal to those originally generated by the expert.

The way in which rules are generated in RuLess is similar to that of the Ripple Down Rules (RDR) method [3] which, like RuLess, is an incremental method for acquiring rules. In both cases, a new rule is produced when a case (an example) is not covered by the existing rules, or when the expert disagrees with the classification given by those rules. RuLess has the advantage over RDR that the set of resulting rules may be simplified without losing the spirit of the original expert's classification strategy.

3 Experiments and Results

The task of obtaining rules from examples is essentially a machine learning (ML) problem. It is therefore reasonable to ask: how does RuLess compare against related ML algorithms? To address this question an experiment to evaluate the performance of RuLess was carried out. In this experiment, the results obtained with RuLess are compared with those obtained with the CN2 induction algorithm [1], which is commonly used as a comparative reference of methods for the generation of rules. The evaluation included the following aspects: (i) the time taken by an expert to develop the rules, (ii) the number of rules generated, (iii) the total number of conditions in the set of rules, (iv) the total number of values in the set of rules, and (v) the comprehensibility of the rules. In the experiment, since different experts may use different strategies for proposing rules, two separate tests were carried out. One set of rules is generated by adopting a strategy where the expert begins defining general rules and then more specific rules later, as required to differenciate new cases from previously classified cases. The second test adopts a strategy where specific rules are generated from the beginning.

The selected order for the attributes and their values was the following.

- The user: novice, experienced.
- The goal of the user: learn/consult, apply.
- The user is familiar with the system: no, yes.
- The importance of the procedure: high, intermediate, low.
- The average difficulty of steps: high, intermediate, low.
- The size of the procedure: big, normal, small.
- The network shape: wide, symmetric, slender.

Each example can be one of the following five classes, where each class is a possible agenda.

- A1: Full Introduction, Concepts & Steps, Examples, Restrictions, Tips, Review and Test.
- A2: Brief Introduction, Concepts & Steps, Examples, Restrictions and Test.
- A3: Brief Introduction, Concepts & Steps, Review and Test.

– A4: Brief Introduction, Examples and Review.
– A5: Brief Introduction and Experimentation.

Table 4 shows five of these examples.

Table 4. Some training examples.

User	Goal	Fam.	Import.	Complex.	Size	Shape	Agenda
novice	learn	no	high	high	big	wide	A1
novice	learn	no	low	interm.	normal	sym.	A2
experienced	apply	yes	low	high	small	wide	A3
experienced	learn	yes	low	interm.	small	wide	A4
novice	learn	no	low	low	small	sym.	A5

The experiments involved the generation of rules for the 648 possible examples from the 7 attributes. The time taken by the expert to generate each set of rules was two hours, approximately. The examples were also used to generate rules using CN2. Both ordered and unordered rules were generated using CN2. The results obtained for each of these 4 sets of rules are presented below.

3.1 First Set of Rules

For the generation of the first set of rules, the expert started defining very general rules. For instance, in the case of the first example (first example in Table 4), the expert defined the following general rule for procedures of high importance.

Rule 1:
If the importance of the procedure is High
then A1: Full Introduction, Concepts & Steps, Examples, Restrictions, Tips, Review and Test.

In total, this first set of rules had originally 42 rules with 213 conditions and 273 values (e.g, rule 27 and 40 shown in Fig. 2 have 5 conditions and 6 values, and 3 conditions and 3 values, respectivelly). The use of the RuLess post-processing mechanism over this set of rules produced a new set of 41 rules with 167 conditions and 180 values. In this case, there were 25 rules which were more specific versions of other rules.

3.2 Second Set of Rules

For the generation of the second set of rules, the expert used a slightly more specific approach from the beginning. For instance, to classify the first example, he used the following rule based on, not only the importance of the procedure, but also on the type of user and the user's goal.

Rule 1:
If the user is Novice
and the goal of the user is Learn/Consult
and the importance of the procedure is High or Intermediate
then A1: Full Introduction, Concepts & Steps, Examples, Restrictions, Tips, Review and Test.

Originally, for this second approach, 53 rules were generated, with 273 conditions and 300 values. By filtering these rules through the RuLess mechanism, a new set of 49 rules with 186 conditions and 199 values was obtained. In this case, there were 17 rules which were more specific versions of other rules.

3.3 Rules from CN2

For this experiment, two sets of rules were generated with CN2: an unordered set and an ordered set. In both cases a Laplacian error estimate, a star size of five, and a significant threshold for accepting rules of 0, where used. The unordered set resulted in 61 rules with 355 simple conditions (one value for each condition), and 3 rules were specific cases of another rule. The ordered set resulted in 47 rules with 116 single conditions, and 35 rules were specific cases of another rule. In the first case, 17 of the examples are incorrectly classified by the resulting set of rules; in the second case, 10 examples are incorrectly classified. This is because CN2 considers the possibility of noise in the examples.

3.4 Discussion

Table 5 summarizes the results obtained.

Table 5. Summary of the sets of rules.

Set of Rules	*Rules*	*Conditions*	*Values*	*Related Rules*	*Accuracy*
RuLess 1st set before filtering	42	213	273	0	100%
RuLess 1st set after filtering	41	167	180	25	100%
RuLess 2nd set before filtering	53	273	300	0	100%
RuLess 2nd set after filtering	49	186	199	17	100%
CN2 unordered set	61	355	355	3	97.4%
CN2 ordered set	47	116	116	35	97.5%

The following conclusions can be drawn from these results.

- The use of the RuLess filtering mechanism produced a notable simplification of the original expert's rules. In both sets the total number of conditions was reduced by a little more than one third.

- The number of rules obtained by defining very general rules (first set), was smaller than those obtained by defining more specific rules from the beginning (second set). This last set has a larger number of independent rules compared to those of the first set. One would expect this since by being more specific, the rules embrace a smaller group of examples (the extreme case would be to have one rule for each example).
- In both sets of rules, RuLess produced fewer and simpler rules than those of the CN2 unordered set.
- The two sets of rules generated with RuLess make a correct classification of all the examples. On the other hand, the two sets of rules generated by CN2 (which consider the possibility of noisy data) produce a small number of incorrect classifications. This could be seen as a tradeoff between simplicity and precision.
- The rules with the fewest conditions are the CN2 ordered rules, but with the inconvenience that they are the less independent (they are closely interwoven) and thus the least comprehensive.
- The more independent rules are the CN2 unordered rules. However, they require an additional mechanism to resolve any rule conflicts that may occur [1], thus detracting from a strict logical interpretation of the rules.
- The rules generated with RuLess can be considered the most comprehensible and maintainable. This is because: (i) they are based on the rules generated by the expert, (ii) although they are ordered, they are relatively few and simple rules, (iv) the relation between related rules is explicitly indicated, and (v) the RuLess mechanisms enables one to know the examples covered by a rule and which examples are affected in their classification when a rule is added, moved, modified or deleted.

.

4 Conclusions

The major conclusions to be made from the material presented in this paper is that RuLess is a useful method for the incremental generation of comprehensible and simple knowledge bases for domains where: (i) there is a finite set of possible examples which have not been generated and from which a set of classification rules need to be produced, (ii) all the attributes are discrete, and (iii) the number of examples is not too large. Some advantages of the method are as follows.

- The generation of examples and rules is relatively easy and fast.
- During the generation of rules, the expert may select her own strategy to classify the examples.
- Each time a new rule is defined by the user, the consistency of the knowledge base is automatically checked.
- Since the generation of examples and the generation of rules are performed at the same time, the effort required by the expert to generate the rules is not much more than that required to do the classification of examples needed

by a machine learning algorithm. This with the advantage that the resulting rules maintain the original expert criteria.
- The expert does not need to worry about generating redundant or unnecessarily specific rules.
- It embodies both, the advantages of the fast generation of rules (twenty to thirty rules per hour), like in Ripple Down Rules, as well as the generation of simple comprehensible rules (without unnecessary redundances), which are obtained using inductive learning techniques like CN2.

References

1. P. Clark and T Niblett. The CN2 induction algorithm. *Machine Learning*, 2:261–283, 1989.
2. P.R. De Buen, S. Vadera, and E. Morales. A multi-functional knowledge based system to learn, apply, and consult procedures. *Expert Systems with Applications*, 15(3/4):265–275, 1998.
3. Y. Mansuri, J. Kim, P. Compton, and C. Sammut. An evaluation of ripple down rules. In *Proceedings of the Australian Workshop on Knowledge Acquisition for Knowledge-Based Systems*, pages 114–132, Pokolbin, Australia, 1991.
4. K. McGraw and K. Harrison-Briggs. *Knowledge Acquisition*. Englewood Cliffs, N.J: Prentice Hall, Inc., 1990.

Applying Reinforcement Learning to Improve MCOE, an Intelligent Learning Environment for Ecology

Daniel Antonio Callegari[1] and Flávio Moreira de Oliveira[1]

[1] Pontifícia Universidade Católica do Rio Grande do Sul - PUCRS
Programa de Pós-Graduação em Ciência da Computação,
Av. Ipiranga 6681 - prédio 16, Porto Alegre, RS - Brazil - 90610-900
{dcalleg, flavio}@inf.pucrs.br

Abstract. In the field of Intelligent Learning Environments, the use of multi-agent systems technology is getting more and more attention, and several Machine Learning techniques are being applied on such systems. One of the most promising techniques is Reinforcement Learning. This paper presents the MCOE system, a learning environment for teaching Ecology to children, describing one of its limitations: the low flexibility on the generation of simulated situations for the students. We propose modifications on its architecture to make it more dynamic and adaptable through the generation of a wider set of natural and adaptation-like effects. Instead of building a complex, more deliberative solution, we propose solving the problem using a decentralized and more reactive solution by using reinforcement learning concepts, while keeping the simplicity of the architecture, an important feature in systems designed for small school applications.

1 Introduction

Intelligent Tutoring Systems and Intelligent Learning Environments constitute models for computer supported teaching-learning systems. According to [2; 3], the use of multiagent systems technology is getting more and more attention in this field, and several Machine Learning techniques are being applied on such systems. One of the most promising techniques is Reinforcement Learning (RL), especially in multiple agents environments.

The Multi Cooperative Environment (MCOE) [2] is an intelligent, cooperative learning environment for teaching Ecology to children, developed over a multiagent architecture. The environment simulates a lake with living elements such as plants, microorganisms and different types of fish. Two students cooperate in order to maintain the ecological equilibrium, playing different roles and sharing experiences.

The current implementation of the system is being well accepted within the teaching community, but there are still a few limitations, such as the low flexibility on the generation of simulated situations for the students. The present work is focused on this issue, introducing machine learning features in some agents of the system, in order to make it more dynamic and also improve the way in which the system can be explored

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 284—293, 2000.

in the pedagogical point of view. Modifications are being introduced on the system's architecture to support Negotiated W-Learning, a multiagent-oriented variation of reinforcement learning, based on competition [5].

In the next section, we describe more precisely the MCOE system and its architecture. The sections 4 and 5, respectively, present the basic concepts of reinforcement learning and Negotiated W-Learning. The section 6 describes the improvements of the system architecture and implementation issues. Section 7 presents the current research state, some concluding remarks and suggestions for future work.

2 The MCOE System

MCOE is a cooperative intelligent learning environment for children that can also be viewed as an educational multimedia game based on ITS technology [2]. The game's environment is an artificial, simplified ecosystem, composed by a lake with living elements such as fish, microorganisms and plants, where pollutant elements are randomly activated on time. Two students choose different characters (Mayor, Citizen, Tourist or Mother-Nature) and cooperate on the goal of keeping the ecological equilibrium. Each character has a distinct set of tools that can be used to fight against the pollution. Each student is free to choose any of its tools and can interact with his/her partner to compose a common strategy to fight pollutant elements. The quality of the ecosystem in terms of ecological equilibrium is expressed by an "ecometer". By observing the students' actions, the tutor component of the system is able to plan its behavior and present information according to each situation. Fig. 1 presents MCOE's interface.

Fig. 1. The MCOE System [2]

The system is composed by a hybrid multiagent architecture integrating reactive and cognitive agents. The cognitive agents are reasoning and learning agents that have

an explicit representation of knowledge, beliefs and goals. The tutor and the students' agents are implemented with cognitive agents (Fig. 2).

The reactive agents have an explicit representation of behavior, where their activities are produced by interactions with the environment. Those agents implement the elements of the environment, including both animals and plants and the pollutant elements. The modeling of fish is presented as an example in Fig. 3.

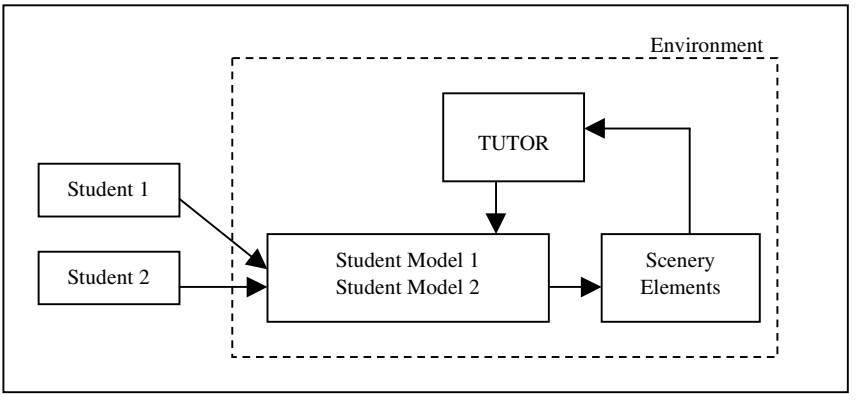


Fig. 2. MCOE's Architecture [2]

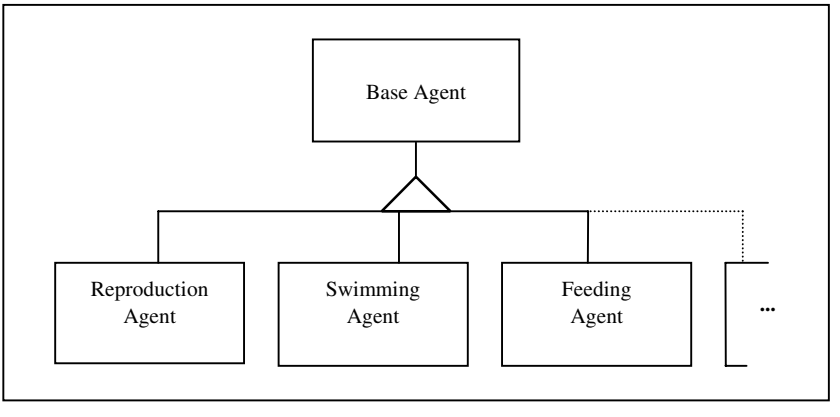


Fig. 3. Modeling fish as reactive agents

In the original implementation, each reactive agent was composed by a base agent, which stored functions and characteristics common to all reactive agents, and a set of subagents for its specific actions in the environment. Since their internal representations of behaviors were explicit and fixed, these agents were not capable of reasoning and learning, thus the number of distinct environments that could be simulated was also fixed. This problem represents a constraint regarding the set of situations the system can provide to the students.

In the next section we present the basics of reinforcement learning, the technology chosen to increase the variety of possible simulation environments, improving the system.

3 Reinforcement Learning

According to [8], the development of a machine learning system involves many choices, including the type of training experiences, the function to be learned and its representation and, moreover, a learning algorithm that learns over the training samples. This traditional approach, although, may become complicated for some systems. As an alternative to planning mechanisms there are methods based on estimates of the best action to be taken in each situation.

Reinforcement learning is a machine learning technique based on the idea that the tendency to produce an action should be strengthened (reinforced) if it produces favorable results, and weakened if it produces unfavorable results [4]. Sutton and Barto define RL as learning what to do (how to map situations to actions) so as to maximize a numerical reward signal [10].

In many cases, it is clear that building a lookup table for instant actions it is not enough. What is important is to get the best sequence of actions on time. The discovery of that sequence may in fact have the same power that planning on traditional systems. As Russell points out, reinforcement learning is still one of the most active areas in machine learning, due to its capacity of eliminating hand-written control strategies [9].

In a reinforcement learning system, the state of the environment is represented by a set of state variables perceived by the agents using sensors. The set of all combinations of these variables is called the agent's state space. An action performed by an agent changes the state of the environment and the value of this transition is passed to the agent as a reinforce signal, referred to as the reward. The agent, thus, has to choose the sequence of actions that tend to increase the sum of reinforce values, i.e., the sequence of actions that produce the highest accumulated reward on time. This is an iterative process, in which the agent moves autonomously on its state space, learning about the environment.

Also, in RL, the agent learns actions by trial and error. Every time it performs an action, the agent receives a reward or penalty indicating how desirable is to reach the resulting state. It is important to note that the reinforced value may not always indicate advances: it may be a negative reinforcement telling the agent to be careful with that action on that state. In reinforcement learning we do not specify how the tasks have to be done. This issue is left to the agents. One great advantage of RL is that the learning is done on-line - there is no need for a explicit training phase.

Any reinforcement learning problem has three fundamental parts: the environment, where the agents live; the reinforcement function, which provides feedback of the actions to the agent; and the value function, a mapping of states to values (or quality) of states. One of the main problems is how to find efficiently the optimum value function [4]. The Q-Learning method is one of the most used methods. In this method, the agent tries to determine Quality-Values (Q-Values) for each (state, action) pair. At every step, the learning mechanism has to update the expression:

$$Q(i, a) \rightarrow (R(i, a) + \gamma \max_{b \in A} Q(j, b))$$

where:

- A is the set of actions;
- "i" is the current state, and "j" the next state;
- Q(i, a) is the value (quality) of action "a" on state "i";
- R(i, a), the reward of action "a" on state "i";
- γ, a discount factor; and
- "Q(i, a) $\rightarrow$ d" means adjust the estimate "Q(i, a)" in direction of "d".

The update function for the Q-Values is the following:

$$Q(i, a) \leftarrow (1-\beta) \cdot Q(a, i) + \beta \cdot (R(i, a) + \gamma \max_{b \in A} Q(j, b))$$

where:

- β is the learning rate; and
- γ, a discount factor.

Briefly, in this update function, the lesser is β, the smoother is the learning. The constant γ represents how the past actions interfere on the current update, i.e., the greater is γ, more the results of the past actions influence the value of the chosen action in the current state. For more details, please see [8; 9; 10].

Many of the problems in reinforcement learning can be seen as action selection problems. By action selection we understand the task of finding out which of the possible actions in a given state is the most appropriate on that situation. In the case of the MCOE system, this same idea can be applied to the agents representing the fish, since, at each step, they must choose actions concerning the task of satisfying multiple objectives (feeding, avoiding predators, etc.) and even conflicting ones (as, for example, when the food and the predator are in the same direction).

Even though Q-Learning is still the most used method, the increasing demand for multiagent systems has created the necessity of extending these techniques so that they could also be efficiently used on systems based on multiple agents. The technique described in the next section is one of the derived methods and was chosen to solve the learning problem in MCOE.

4 Negotiated W-Learning

In a standard reinforcement learning system we have typically a single agent or process being responsible for the whole task of learning on its environment. However, there are problems where a decentralized solution is more adequate: action selection in the presence of multiple goals is a typical case of this situation, since it is possible to learn strategies for each goal individually and then combine them according to the agent's current state.

In the words of Connell & Mahadevan, "decomposing a huge learning problem into a collection of smaller ones, and providing useful reinforcement signals for the subproblems is a very powerful technique for biasing learning. Most interesting examples of robotic reinforcement learning employ this technique to some extent" (Connell & Mahadevan, 1993 in [6]).

In the case of MCOE, we adopted a learning method derived from the traditional monolithical reinforcement learning technique [11]. This new method, called Negotiated W-Learning (NWL), is a decentralized version of the Q-Learning algorithm, in which we have several independent agents connected to a main agent (the switch) that determines which of them must be attended at every step.

Basically, the idea behind NWL comes from Hierarchical Q-Learning, where we break up a complex problem into sub-problems, having a collection of Q-Learning agents learning the sub-problems and a single controller Q-Learning agent (the switch) which learns Q(x, i), where "i" is which agent to choose in state "x" [7].

In this model, we literally have a competition among agents that react to different stimuli. This save us from building a complex cooperative mechanism involving all the necessary parts, including communication and negotiation. More than that, every subagent senses or perceives only a subset of the complete state information (a subset of the state variables). This contributes directly to the performance of the system, reducing the risk of combinatorial explosion.

Below is presented the algorithm executed by the switch. At every step, the agent observes the current state and each subagent s_i, based on its partial perceptions, suggest an action a_i. The switch, then, chooses a winner k and executes action a_k.

```
Switch()
{
        observe state x

        start with leader k := (random agent s),
            and Wk := 0;

        loop:
            for all agents i, other than k
                Wi := Qi(x, ai) - Qi(x, ak);
            if (highest Wi) > Wk
                // (found new leader)
                goto loop;

        // (loop has terminated with winner k)

        execute ak;
}
```

The NWL algorithm is being applied to a collection of fish in MCOE. Their goal is learn to choose the right actions in every observed state. They must avoid predators, hunt smaller fish, and be able to swim to safer areas to reproduce or run away from the pollution. Figure 4 presents the architecture model of NWL.

According to Humphrys, this algorithm discovers explicitly in one timestep what traditional W-Learning only learns over time. For a more detailed description, please see [5].

We now present the modifications that are being introduced to the system.

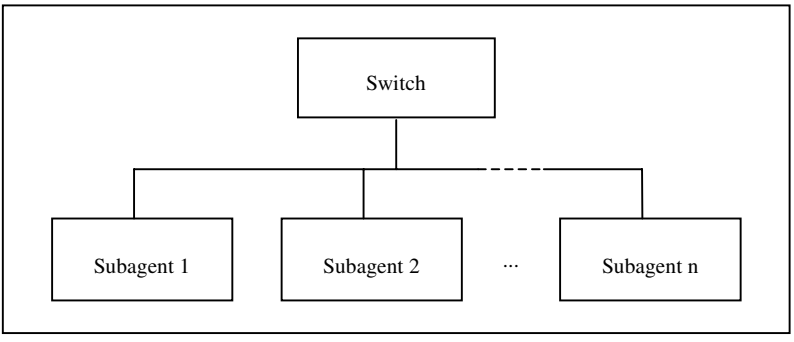


Fig. 4. Architecture model for NWL-based agents

5 Extensions to MCOE's Architecture

The first modification on the system was realized on the architecture of the reactive agents. A new structure of classes was developed in C++, joining the necessary components to the learning algorithm. Based on NWL, we defined the model for the fish as exposed on Fig. 5. Each individual has its own instance of the structure and, thus, learns independently of the others.

It is important to mention that a subagent does not "know" that the others exist and that it is participating on a competition for the switch's attention. This provides a great advantage over other techniques because there is no need for reconfigurations when adding or removing subagents, or even communication between them. As we mentioned before, each subagent senses only what is important to make its decisions. Thus, for instance, the "Avoid Predators" subagent only takes into account its relative distance to the nearest visible predator and its food level. The remaining subagents work similarly.

At every simulation step, each subagent activates its sensors identifying the current state, then updates its knowledge and chooses an action to be executed (since any subagent can be pointed out as the winner by the switch). After that, the switch starts the competition loop for that step and the winner gets the right to execute its previously chosen action. Naturally, two or more subagents may choose contradictory actions, but this will be solved by the competition itself.

Another important fact is that the set of possible actions is the same for all subagents (this is required by the NWL algorithm). The actions correspond to make moves in one of the allowed directions for the agent in the environment, observing possible collisions with surface, the borders and the bottom of the lake (the positions in the environment are mapped to a grid).

In order to learn, the subagents must be able to perceive and evaluate the results of their actions. Thus, every time a subagent executes an action, it receives a feedback (reward) according to a set of explicit criteria. For instance, being able to escape from a predator results in a positive reward, while remaining in a polluted area results in a

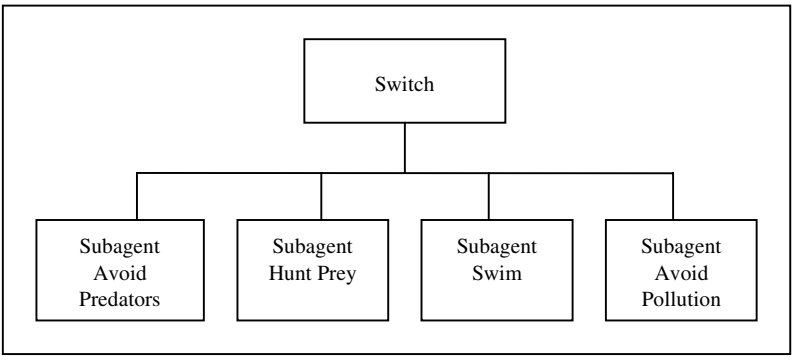


Fig. 5. NWL-based architecture model for the fish

negative reward (penalty). As the simulation goes, the agents learn and adapt their behavior to survive in the environment.

The feedback to the learning agents is given through numerical rewards between -1.0 and 1.0. The "Avoid Predators" subagent, for example, has three main states: (a) no predator visible; (b) at least one predator visible; and (c) captured (actually there are more states, but here we will work with only a few for simplicity). The rewards are given every time a change on states occurs. An example of a reward function is presented in figure 6.

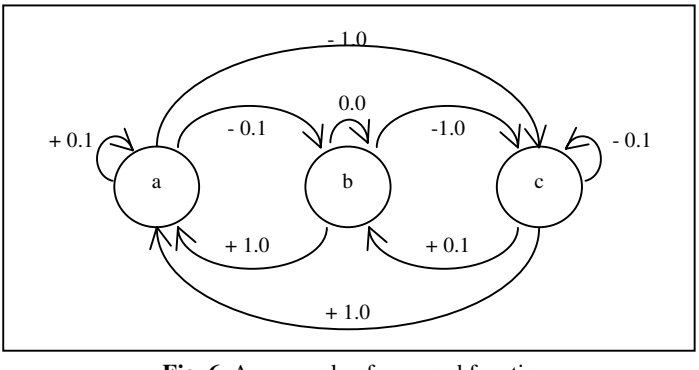


Fig. 6. An example of a reward function

In Figure 6 we see the states and the rewards attached to each state transition. When an agent moves from a dangerous position (a predator was near) to a safer position (no predator visible), i.e., when a (b) to (a) state transition happens, it has just escaped from a predator, so it receives a positive reinforcement of 1.0. If, for instance, the agent executes an action that takes it again to a dangerous position, i.e., a transition from (a) to (b), it receives a penalty of 0.1. All the remaining subagents are designed in the same form. Each one has its own set of states and reward function, giving a total of 33 states per individual [1].

Another example of a reward function is presented below (Fig. 7). This reward function is used by the “Hunt Prey” subagent. There are four main states (similar states were condensed to states that present “Px”, which stands for “prey visible”). All

states are combinations of the situations “hungry / not hungry”, “prey visible / no preys visible” and “prey captured”. The rewards represent how desirable is to go from one state to another. For instance, all transitions to the F0Px state (not hungry) return a positive reward of +1.0, while a transition from F1Px to F1P0 (the prey escaped) returns a negative reinforcement of –1.0.

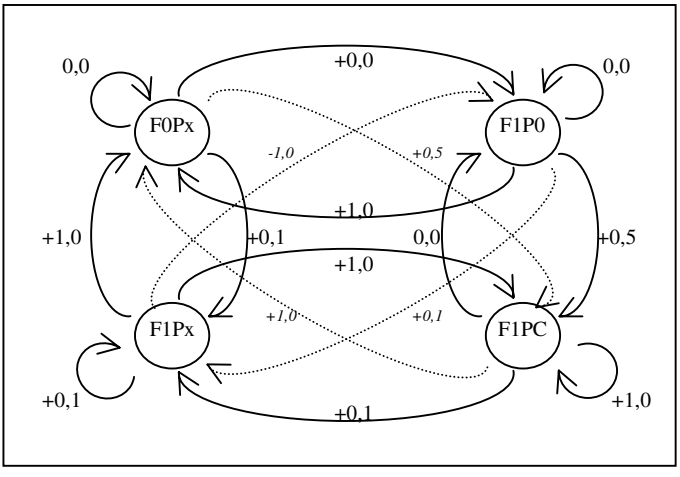


Fig. 7. The reward function for the Hunt Prey subagent

The Q-Values in reinforcement learning are normally stored in a state/action matrix or represented by a function approximator such as a neural network, depending on its complexity. In our case, a few matrices (one for each subagent) are enough, since the number of states and actions is not large.

We performed several experiments, varying parameters as the learning rate, the exploration threshold, the discount factor and the number of agents on each run. The obtained results are very encouraging and their full description can be found in [1].

The previous version of MCOE presented a few limitations. With the introduction of the NWL algorithm and the proposed model, we added to the fish the capacity of perceiving changes in the environment, being able to adapt their behavior gradually on time.

5 Final Remarks and Future Work

This paper described the MCOE system and the modifications on its architecture in order to make it more dynamic and adaptable through the generation of a wider set of natural and adaptation-like effects. The fish in MCOE are a typical case of action selection problem, where there are multiple and conflicting goals to be satisfied simultaneously. Instead of building a complex, more deliberative, coordinated or plan-based solution, we proposed solving the problem using a decentralized and more reactive solution using reinforcement learning concepts.

Through a simple architecture we saw that it is possible to get a significative improvement in the behavior of the agents. The simplicity of the implementation is important in MCOE since it was designed for low-age children and for low-cost machines in schools.

The prototype was developed in C++ and written in portable code, and the learning mechanism is being optimized through iterative refinements. Since our aim is not to build "optimal" fish, but to keep them always adapting to the changes in the environment, the results so far obtained show that the introduction of learning abilities to the fish allows to constantly provide new experiences to the students.

The architecture's modularity allows us to make the modifications on the reactive part of the system separately from the cognitive part. In fact, it is clear that the knowledge of the tutor agent must be adapted to the new parameters. These adjustments are good candidates for future work on the area, adding to the tutor the capacity of taking account of the changes in the behavior of the improved reactive agents.

References

1. Callegari, Daniel A.: Aplicando Aprendizagem por Reforço a uma Arquitetura Multiagente para Suporte ao Ensino de Ecologia. (Master Thesis), Porto Alegre, Brazil, PUCRS, 2000
2. Giraffa, Lucia M. M.: Uma Arquitetura de Tutor Utilizando Estados Mentais. (PhD Thesis), Porto Alegre, Brazil, PGCC/UFRGS, 1999
3. Gürer, Denise.: The Use of Distributed Agents in Intelligent Tutoring. ITS'98 Workshop on Pedagogical Agents. USA, 1998
4. Harmon, Mance E.: Reinforcement Learning: A Tutorial. Available at http://www-anw.cs.umass.edu/~mharmon/rltutorial, 1999
5. Humphrys, Mark.: Action Selection methods using Reinforcement Learning. (PhD Thesis), Cambridge, 1997
6. Kaelbling, Leslie Pack; Littman, Michael L.; Moore Andrew W.: Reinforcement Learning: A Survey. Available at: http://www.cs.washington.edu/research/jair/abstracts/ kaelbling96a.html, 1996
7. Long-Ji, Lin.: Scaling up Reinforcement Learning for Robot Control. Proceedings of the Tenth International Conference on Machine Learning, 1993
8. Mitchell, Tom M.: Machine Learning. McGraw-Hill, USA, 1997
9. Russell, Stuart J; Norvig, Peter.: Artificial Intelligence: A Modern Approach. Prentice-Hall, USA, 1996
10. Sutton, Richard; Barto, Andrew.: Reinforcement Learning: An Introduction. MIT, USA, 1998
11. Watkins, Christofer.: Learning From Delayed Rewards. Cambridge, 1989

Rapid Fine-Tuning of Computationally Intensive Classifiers

Stefan Zemke

Department of Computer and Systems Sciences
Stockholm University/KTH, Electrum 230, 164 40 Kista, Sweden
Stefan.Zemke@dsv.su.se

Abstract. This paper proposes a method for testing multiple parameter settings in one experiment, thus saving on computation-time. This is possible by simultaneously tracing processing for a number of parameters and, instead of one, generating many results – for all the variants. The multiple data can then be analyzed in a number of ways, such as by the binomial test used here for superior parameters detection. This experimental approach might be of interest to practitioners developing classifiers and fine-tuning them for particular applications, or in cases when testing is computationally intensive.

1 Introduction

Evaluating a classifier and fine-tuning its parameters, especially when performed with non-optimal prototype code, all often require lengthy computation. This paper addresses the issue of such experiments, proposing a scheme speeding up the process in two ways: by allowing multiple classifier-variants comparison in shorter time, and by speeding up detection of superior parameter values.

The rest of the paper is organized as follows. First, a methodology of comparing classifiers is described pointing out some pitfalls. Next, the proposed method is outlined. And finally, an application of the scheme to a real case is presented.

2 Basic Experimental Statistics

2.1 Comparing Outcomes

While testing 2 classifiers, one comes with 2 sets of resulting accuracies. The question is then: are the observed differences indicating actual superiority of one approach or could they arise randomly.

The standard statistical treatment for comparing 2 populations, the t-test, came under criticism when applied in the machine learning settings [2], or with multiple algorithms [3]. The test assumes that the 2 samples are independent, whereas usually when two algorithms are compared, this is done on the same data set so the independence of the resulting accuracies is not strict. Another doubt can arise when the quantities compared do not necessarily have normal distribution.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 294–302, 2000.

If one wants to compare two algorithms, A and B, then the binomial test is more appropriate. The experiment is to run both algorithms N times and to count the S times A was better than B. If the algorithms were equal, i.e., *P(A better than B in a single trial) = 0.5*, then the probability of obtaining the difference of S or more amounts to the sum of binomial trials, $P = 0.5$, yielding between S and N successes. As S gets larger than $N/2$, the error of wrongly declaring A as better than B decreases, allowing one to achieve desired confidence level. Figure 1 provides the minimal S differentials as a function of number of trials N and the (I or II sided) confidence level.

#Trials	95% I	95% II	99% I	99% II	99.9% I	99.9% II	99.99% I	99.99% II
5	5	-	-	-	-	-	-	-
6	6	6	-	-	-	-	-	-
7	7	7	7	-	-	-	-	-
8	7	8	8	8	-	-	-	-
16	12	13	13	14	15	15	16	16
32	22	22	23	24	25	26	27	28
64	40	41	42	43	45	46	47	48
128	74	76	78	79	82	83	86	87
256	142	145	147	149	153	155	158	160
512	275	279	283	286	292	294	299	301
1024	539	544	550	554	562	565	572	575

Fig. 1. Minimal success differentials for desired confidence

The weaknesses of binomial tests for accuracies include: non-qualitative comparison – not visualizing how much one case is better than the other (e.g., as presented by their means), somehow ambivalent results in the case of many draws – what if the number of draws $D >> S$, should the relatively small number of successes decide which sample is superior, non-obvious ways of comparing more than 2 samples or samples of different cardinality [4].

2.2 Significance Level

Performing many experiments increases the odds that one will find 'significant' results where there is none. For example, an experiment at 95% confidence level draws a conclusion that is with 0.05 probability wrong, so in fact, for every 20 experiments, one is *expected* to pass arbitrary test with 95% confidence. The probability of not making such an error in all among K (independent) experiments goes down to 0.95^K, which for $K > 1$ is clearly less than the 95% confidence level.

Thus in order to keep the overall confidence for a series of experiments, the individual confidences must be more stringent. If c is the desired confidence, then the productive confidence of the individual experiments must be at least

that. Figure 2 presents, for few desired levels, maximally how many experiments at a higher level can be run for the series to still be within the intended level. The approximate (conservative) formula is quite simple *MaxNumberOfTrials = (1 - Confidence_desired) / (1 - Confidence_tested).*

Confidence desired	95%	99%	99.9%
tested			
99%	5	1	-
99.9%	51	10	1
99.99%	512	100	10

Fig. 2. Required single-trial confidence for series of trials

To avoid spurious inferences, one is strongly advised to always aim at significance higher than the bottom line 95% easily obtained in tens of testing runs. However, more stringent tests also increase the possibility that one will omit some genuine regularities. One solution to this trade-off could be, first searching for any results accepting relatively low significance, and once something interesting is spotted, to rerun the test, on a more extensive data, aiming at a higher pass.

2.3 Tuning Parameters

A common practice involves multiple experiments in order to fine-tune optimal parameters for the final trial. Such a practice increases the chances of finding an illusory significance – in two ways. First, it involves the discussed above effect of numerous tests on the same data. Second, it specializes the algorithm to perform on the (type of) data on which it is later tested.

To avoid this pitfall, first each fine-tuning experiment involving the whole data should appropriately adjust the significance level of the whole series – in a way discussed. The second possibility requires keeping part of the data for testing and never using it at the fine-tuning stage, in which case the significance level must only be adjusted according to the number of trials on the test portion.

3 The Method

Usually it is unclear without a trial how to set parameter values for optimal performance. Finding the settings is often done in a change-and-test manner, which is computationally intensive, both to check the many possible settings, and to get results enough as to be confident that any observed regularity is not merely accidental. The proposed approach to implementing the change-and-test routine can speed up both.

The key idea is to run many experiments simultaneously. For example, if the tuned algorithm has 3 binary parameters A, B and C taking values -/+, in

order to decide which setting among A- B- C-, A- B- C+, ..., A+ B+ C+ to choose, all could be tried at once. This can be done by keeping 2 copies of all the variables influenced by parameter A: one variable set representing the setting A- and the other – A+. Those 2 variable sets could be also used in 2 ways – each with respect to processing required by B- and B+ resulting in 4 variable sets representing the choices A- B-, A- B+, A+ B- and A+ B+. And in the same manner, the C choice would generate 8 sets of affected variables. Finally, as the original algorithm produces one result, the modified multiple-variable version would produce 8 values per iteration.

The details of the procedure, namely which variables need to be traced in multiple copies, depend on the algorithm in question. Though the process might seem changing the structure of the algorithm – using data structure in the place of a single variable – once this step is properly implemented, it does not increase the conceptual complexity if 2 or 10 variables are traced. Actually, with the use of any programming language allowing abstractions, such as an object-oriented language, it is easy to reveal the internal nature of variables only where necessary - without the need for any major code changes where the modified variables are merely passed.

Handling the variable choices obviously increases the computational complexity of the algorithm, however, as it will be shown on an example, the overhead can be negligible when the variable parameters concern choices outside the computationally intensive core of the algorithm, as it usually is in the case for fine-tuning[1].

3.1 Superior Parameter Detection

Continuing the above case with 3 binary choices, for each classifier application 8 outcomes would be generated instead of one, if all the 3 parameters were fixed. Concentrating on just one parameter, say A, divides the 8 outcomes into 2 sets: this with A- and A+ – each including 4 elements indexed by variants of the other parameters: B- C-, B- C+, B+ C-, B+ C+. The identical settings for the other parameters allow us to observe the influence of value of A by comparing the corresponding outcomes.

The comparisons can be made according to the binomial test, as discussed [4]. In order to collect the statistics, several iterations – applications of the algorithm – will usually be required, depending on the number of variable choices – so outcomes – at each iteration, and the required confidence. With 3 variable choices, each application allows 4 comparisons – in general, tracing K choices allows 2^{K-1}.

This analysis can reveal if a certain parameter setting results in significantly better performance. The same procedure, and algorithm outcomes, can be used for all the parameters, here including also B and C, which equally divide the outcomes into B- and B+, etc. Any decisive results obtained in such a way indicate

[1] It is matter of terminology what constitutes parameter-tuning and what development of a new algorithm.

a strong superiority of a given parameter value – regardless of the combinations of the other parameters. However, in many cases the results cannot be expected to be so crisp – with the influence of parameter values inter-dependent, i.e. which given parameter value is optimal may depend on the configuration of the other parameters.

In that case the procedure can be extended, namely the algorithm outcomes can be divided according to value of a variable parameter, let it be A, into 2 sets: A- and A+. Each of the sets would then be subject to the procedure described above, with the already fixed parameter excluded. So the analysis of the set A- might, for example, reveal that parameter B+ gives superior results no matter what the value of the other parameters (here: only C left), whereas analysis of A+ might possibly reveal superiority of B-. The point to observe is that fixing one binary variable reduces the cardinality of the sample by half, thus twice as many algorithm iterations will be required for the same cardinality of the analyzed sets. This kind of analysis might reveal the more subtle interactions between parameters, helpful in understanding why the algorithms works the way it does.

3.2 Parallel Experiments

In the limit, the extended procedure will lead to 2^K sets obtaining one element per iteration, K – the number of binary parameters traced. Such obtained sets can be subject to another statistical analysis, this time the gains in computation coming from the fact that once generated, the 2^K sets can be compared to a designated set, or even pair-wise, corresponding to many experiments.

The statistics used in this case can again involve the binomial comparison or – unlike in the previous case – a test based on random sampling. In the superior parameter detection mode, the divisions obtained for a single parameter most likely do not have normal distribution, thus tests assuming it, such as the t-test, are not applicable. Since the binomial test does not make any such assumption it was used.

However, if the compared sets are built in one-element-per-iteration fashion, where each iteration is assumed to be independent (or random generator dependent) from the previous one, the sets can be considered random samples. The fact that they are originating from the same random generator sequence forming the outcomes at each iteration, can be actually considered helpful in getting more reliable comparison of the sets – due only to the performance of the variants, but not to the variation in the sampling procedure. This aspect could be considered another advantage of the parallel experiments. However, discussing the more advanced tests utilizing this property is beyond the scope of the current paper.

4 Example of an Actual Application

This section provides description of a classifier development [5], which inspired the parameter tuning and testing procedure. Since the developed algorithm was

(believed to be) novel, there were no clear guidelines which, among the many, small but important choices within the algorithm should be preferred. By providing with more results for analysis, the testing proposed approach helped both to find promising parameters and to clarify some misconceptions about the algorithm's performance. Generating the data took approximately one week of computation, thus repeating the run for the 13 variants considered would be impractical.

4.1 Tuned Algorithm

The designed classifier was an extension of the nearest neighbor algorithm, with parameters indicating what constitutes a neighbor, which features to look at, how to combine neighbor classifications etc. The parameters were optimized by a genetic algorithm (GA) whose population explored their combinations. The idea believed to be novel, involved taking – instead of the best GA-evolved classifier – part of the final GA-population and bagging [1] the individual classifiers together into an ensemble classifier. Trying the idea seemed worthwhile since bagging is known to increase accuracy benefiting from the variation in the ensemble – exactly what a (not over-converged) GA-population should offer.

The computationally intensive part was the GA search – evolving a population of parameterized classifiers and evaluating them. This had to be done no matter if one was interested just in the best classifier or in a bigger portion of the population. As proposed, the tested algorithm needs to be multi-variant traced for a number of iterations. Here, iteration involved a fresh GA run, and yielded accuracies (on the test set) – one for each variant traced.

The questions concerning bagging the GA population involved: which individual classifiers to bag – all above-random or only some of them, how to weight their vote – by single vote or according to accuracy of the classifiers, how to solicit the bagged vote – by simple majority or if the majority was above a threshold. The questions gave rise to 3 parameters and their $3 * 2 * 2 = 12$ value combinations, enumerated in the table below, indicating which parameter number (No) takes what value (+). The parameters, concisely explained in Figure 3, are the following.

1. This parameter takes 3 values depending which of the above-random (fitness) accuracy classifiers from the final GA population are included in the bagged classifier: all, only the upper half, or a random half among the above-random.
2. This binary parameter distinguishes between unweighted vote (+): where each classifier adds 1 to its class, and a weighted vote (-): where the class vote is incremented according to the classifier's accuracy.
3. This binary parameter decides how the bagged ensemble decision is reached – by taking the class with the biggest cumulative vote (+), or (-) when the majority is by more than $\frac{1}{3}$ total votes greater than that of the next class and returning the bias of the training data otherwise.

No	Parameter setting	1	2	3	4	5	6	7	8	9	10	11	12
1	Upper half bagging	+	+	+	+	-	-	-	-	-	-	-	-
1	All above-random bagging	-	-	-	-	+	+	+	+	-	-	-	-
1	Half above-random bagging	-	-	-	-	-	-	-	-	+	+	+	+
2	Unweighted vote	+	+	-	-	+	+	-	-	+	+	-	-
3	Majority decision	+	-	+	-	+	-	+	-	+	-	+	-

Fig. 3. Settings for 12 parameter combinations.

4.2 Parameter Analysis

The parameter analysis can identify algorithm settings that give superior performance, so they can be set to these values. The first parameter has 3 values which can be dealt with by checking if results for one of the values are superior to both of the others. The figure below presents the comparisons as probabilities for erroneously deciding superiority of the left parameter set versus one on the right. Thus, for example, in the first row comparison of {1..4 } vs. {5..8}, which represent the different settings for parameter 1, the error 0.965 by 128 iterations indicates that setting {1..4 } is unlikely to be better than {5..8}. Looking at it the other way: {5..8} is more likely to be better than {1..4 } with error[2] around $0.035 = 1 - 0.965$. The setting {0} stands for results by the reference non-bagged classifier – respective GA run fittest. Figure 4 presents the results.

The following conclusions are for results up to 128 iterations, the results for the full trials up to 361 iterations included for comparison only.

1. There is no superior value for parameter 1 – such that it would outperform all the other values.
2. Both settings for parameter 2 are comparable.
3. Majority decision ({1,3,5,7,9,11}), for parameter 3, is clearly outperforming with confidence 99.99% by 64 iterations.
4. In the comparisons against the non-bagged {0}, settings 5 and 9 are more accurate, at less than 0.1% error (by iteration 128) pointing out superior parameter values.

4.3 Speed up

In this case the speed up of the aggregate experiments – as opposed to individual pair-wise comparisons – comes from the fact that the most computationally

[2] The error probabilities of A- vs. A+ and A+ vs. A- do not add exactly to 1 for two reasons. First, draws are possible thus the former situation S successes out of N trials can lead to less than $F = N - S$ successes for the later, so adding only the non-draw binomial probabilities would amount for less than 1. And second, even if there are no draws, both error binomial sums would involve a common factor $binomial(N, S) = binomial(N, N - S)$ making the complementary probabilities to add to more than 1. Thus for the analysis to be strict, the opposite situation error should be computed from the scratch.

No	Parameter settings/Iterations	32	64	128	361
1	{1..4 } vs. {5..8}	0.46	0.95	0.965	0.9985
1	{1..4 } vs. {9..12 }	0.53	0.75	0.90	0.46
1	{5..8 } vs. {9..12 }	0.33	0.29	0.77	0.099
2	{1,2,5,6,9,10} vs. {3,4,7,8,11,12}	0.53	0.6	0.24	0.72
3	{1,3,5,7,9,11} vs. {2,4,6,8,10,12}	0.018	9E-5	3E-5	0
-	{1} vs. {0}	0.0035	0.0041	0.013	1E-6
-	{2} vs. {0}	0.19	0.54	0.46	0.91
-	{3} vs. {0}	0.055	0.45	0.39	0.086
-	{4} vs. {0}	0.11	0.19	0.33	0.12
-	{5} vs. {0}	0.0035	3.8E-5	6.2E-5	0
-	{6} vs. {0}	0.30	0.64	0.87	0.89
-	{7} vs. {0}	0.055	0.030	0.013	1.7E-4
-	{8} vs. {0}	0.025	0.030	0.02	0.0011
-	{9} vs. {0}	0.055	7.8E-4	2.5E-4	0
-	{10} vs. {0}	0.11	0.35	0.39	0.73
-	{11} vs. {0}	0.19	0.64	0.39	0.085
-	{12} vs. {0}	0.055	0.030	0.0030	0.0016

Fig. 4. Experimental parameter setting comparisons.

intensive part of the classification algorithm – the GA run – does not involve the multiply-threaded variables. They come into play only when the GA evolution is finished and different modes of bagging and non-bagging are evaluated.

Exploring variants outside the inner loop can still benefit algorithms in which multiple threading will have to be added to the loop thus increasing the computational burden. In this case, the cost of exploring the core variants should be fully utilized by carefully analyzing the influence of the (many) post-core settings as not to waste the core computation due to some unfortunate parameter choice afterwards.

5 Conclusion

This paper proposes a method for testing multiple parameter settings in one experiment, thus saving on computation-time. This is possible by simultaneously tracing processing for a number of parameters and, instead of one, generating many results – for all the variants. The multiple data can then be analyzed in a number of ways, such as by the binomial test used here for superior parameters detection. This experimental approach might be of interest to practitioners developing classifiers and fine-tuning them for particular applications, or in cases when testing is computationally intensive.

The current approach could be refined in a number of ways. First, finer statistical framework could be provided taking advantage of the specific features of the data generating process, thus providing crisper tests, possibly at smaller sample size. Second, some standard procedures for dealing with common classifiers could be elaborated, making the proposed development process more straightforward.

References

1. Breiman, L.: Bagging Predictors. Machine Learning 24 (1996) 123-140
2. Dietterich, T.: Statistical Tests for Comparing Supervised Learning Algorithms. Technical Report, Oregon State University, Corvallis, OR (1996)
3. Raftery, A.: Bayesian Model Selection in Social Research. In: Marsden, P. (ed.), Sociological Methodology. Blackwells, Oxford, UK (1995) 111-196
4. Salzberg, S.: On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach. Data Mining and Knowledge Discovery 1 (1997) 317-327
5. Zemke, S.: Amalgamation of Genetic Selection and Bagging. GECCO-99 Poster, http://www.genetic-algorithm.org/GECCO1999/phd-www.html (1999)

Qualitative Knowledge Acquisition for Learning Organizations

Rafael E. Bourguet[1] and Rogelio Soto[2]

[1] Department of Systems Engineering, ITESM Campus Monterrey,
Av. Garza Sada 2501 Sur, 64849 Monterrey, NL, México
bourget@cia.mty.itesm.mx
http://www-cia.mty.itesm.mx/~bourget/index.html
[2] Center for Artificial Intelligence, ITESM Campus Monterrey,
Av. Garza Sada 2501 Sur, 64849 Monterrey, NL, México
rsoto@ieee.org
http://www-cia.mty.itesm.mx/~rsoto/index.html

Abstract. This paper addresses the first step about collecting qualitative knowledge from persons working for learning organizations. Mental models of policymakers are considered the sources of knowledge. The main idea is to represent their qualitative knowledge into a computer simulation model. System Dynamics methodology is used. Results of the process carried out with a team of two consultants on a hotel system are presented. Conclusions are derived and future research is discussed to construct knowledge-based systems that incorporate simulation models for learning purposes in business organizations.

1 Introduction

Qualitative knowledge is the softest part in human organizations and the hardest part to manage. It is the knowledge gathered by experience which most of the times is not registered in papers, databases or other storage media. It is just contained in our mental models, in the "back of our heads".

Economic models based on knowledge have increased the demand for tools to elicit and disseminate best practices of the organizations. Practitioners estimate that more than 80% of the knowledge in organizations has qualitative characteristics.

The problem is how to acquire this knowledge that in many of the times the owner is not even aware of it. The issue becomes more challenging when the knowledge is about dynamics of complex systems, that is, how our social, financial, technological and ecological systems evolve through time. Soft laws seem to regulate their behavior according our perceptions. Scientific method is applied scarcely by generating new knowledge in this systems and common sense and intuition of our mental models are more used to make decisions in each case, instead.

The motivation of this research is to find a method to help decision-makers to accelerate their learning process in complex environments using intelligent systems. The

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 303—314, 2000.

method should involve three basic steps of intelligence: collecting, analyzing, and disseminating relevant data and information to the organization [16].

This paper addresses the first step about collecting qualitative knowledge. Sources are considered the mental models of policymakers and their knowledge is represented in computer simulation models. System dynamics methodology is used for the knowledge acquisition in a Learning Organization.

2 Definitions and Context

System Dynamics is the discipline that deals with the changes through the time of the important things of each person or organization. It takes advantage of computer simulation model to understand their behavior [2].

A Learning Organization is "an organization that is continually expanding its capacity to create its future" [15].

Knowledge is the growing capacity to respond effectively to a perceived reality to attain a goal.

Knowledge acquisition consists of two steps: (1) elicitation of the knowledge from the sources and (2) representation of this knowledge in a computer [4]. A useful point of view to acquire knowledge has come from the area of Business and Administration called Modeling for Learning Organizations.

Modeling for Learning Organizations is a "modern" view of modeling that involves the processes of building, using and learning from models. It assumes that models should capture the knowledge and mental data of policymakers, and should blend qualitative mapping with friendly algebra and simulation. The purpose of the models is to support team reasoning and learning, they have to encourage system thinking and scenarios planning. Simulations are a very important part, as they provide consistent stories about the future, but not predictions, which has been the traditional use. The modelers act as facilitators, as knowledge engineers, which design and lead group processes to capture team knowledge. The modelers are who design and delivers learning laboratories that embed models in an overall learning context of group experimentation and dialogue [17].

Four approaches, from Artificial Intelligence and one from Business and Administration, are close related to the problem of qualitative knowledge acquisition: Cognitive Maps (FCM) [7], [8], [11]; Knowledge-Based Simulation (KBS) [12], [14], [19]; Model Base Reasoning (MBR) [1], [13]; Qualitative Simulation (QS) [6], [10], [18] and System Dynamics (SD) [2], [3].

3 Knowledge Acquisition Approach

The approach is based on System Dynamics discipline, and on the hypothesis that knowledge transmission is carried out by Myths, Symbols and Numbers in every culture. It consists, see Fig. 1, of three levels of knowledge acquisition: (1) mapping of

high level (Myths) is constituted by mental models, verbal models, and causal diagrams; (2) mapping of intermediate level (Symbols), by rates of flows and level diagrams, along with a graphical computer language described in Section 4, and (3) mapping of low level (Numbers) is constituted by algebraic and difference equations.

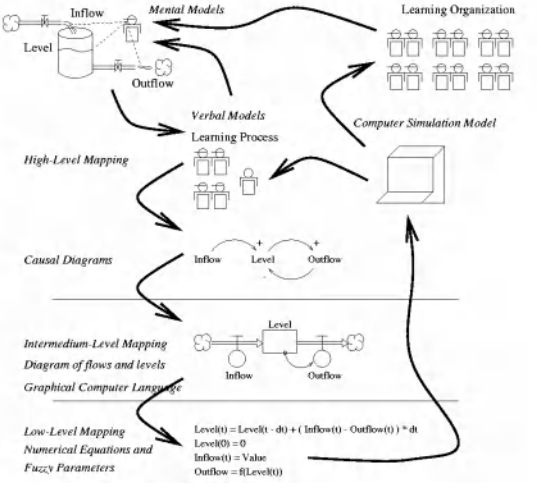


Fig. 1. Process flow diagram

Mapping of high level: mental models, verbal models, and causal loop diagrams. It is characterized by transforming mental models to verbal models and then to causal diagrams. The participants learn throughout the whole process. *Mental models* are those that are based on intuition and experience "in the back of our heads" [9]. Developing these models helps us to learn how to drive a car or how to manage a business. Mental models can be developed by training and experience. *Verbal models* are those used to describe the behavior of a system under different conditions in words: "if the inflation rate goes up then the bank rate will rise". Expert systems are considered instances of formalized verbal models. It is not an easy task to convert a mental model to a verbal model. *Causal diagrams* are important instruments to carry out this task. Cause-effect, causal or influence diagrams are names that are used to mean the same thing in the context of System Dynamics. They are hypothetically correlated set of relationships, focusing on the feedback linkages among elements of a system. The causal diagram represents the story of how the things work. The process through time or evolution of the system is verbally expressed and drawn in a paper. This operation facilitates and gives shape to discussions among participants at this stage. Basically, no questionnaires are used to gather information from managers. They are just asked to relate the stories in a manner of cause-effect relations.

At this level, the main point sharpens and narrows the intended use or purpose of the model. Also, it is the first attempt to divide the system into subsystems in an aggregate form. Efforts are focused to determine cause and effect relationships. This level demands the modeler a large amount of understanding of and intuition for the system to be described. Hierarchies of complexity and approximations are determined.

The first step is to represent the central loop. Questions made to gather information on the central process are on what the nature of the process is or how the process actually works. It should not be asked "What are all the factors that influence the process?" [5].

Mapping of intermediate level: flow rate and level diagrams, and graphic computer language. At the intermediate level mapping, the causal diagrams are transformed into a flow rate and level diagram using a graphical computer language. A rate of flows and level diagram is a computer-graphical description of the state variable representation of the system. The state variable representation allows the study of nonlinear and time varying dynamic systems. Even when there is no way for obtaining an explicit analytic solution for the equation that is describing the system, a realization can be found and a simulation of the system can be obtained. Then the behavior and the cause-effect relations can be observed and analyzed. This characteristic is an advantage for representing complex systems.

The main challenge at this stage is to determine the state variables of the qualitative system description. Physically, a state variable can be associated with a physical storage element. However, for more abstract representations this is not necessary true and a state variable become purely abstract storage but still a storage. A generality of the state variables is that they are always the outputs of integrator blocks. For example, the state variable can be associated with the volume of water in a tank but more practically with its level. Level is an easier variable to observe and measure. Since integration implies accumulation, the accumulation of water in the tank is inferred by the level, and in this way is an integrated variable. In a similar manner the rate of change of the state variables is related to the rate of flow of water that comes into the tank. Tanks can be seen as the integrator blocks, levels as the states variables, and rates of flow as the rates of change of the state variables. One big advantage of using this state-space description is that initial values of the levels provide the minimal sufficient statistic information to calculate the future response of the system for a new input without worrying about the past.

Mapping of low level: numerical equations. The numerical equations are obtained from the flow rate and level diagram. Algebraic and difference first order equations conform now the description of the system. The mapping of low level is carried out by *ithink* software, used in this research.

The following stage is the most challenging task for the modeler and consists on identifying the system heuristically. It is carried out using the most of the available information resources.

4 Graphic Computer Language

The roots of the following graphical computer language come from the conceptual framework and methodology of System Dynamics. Five graphical symbols conform the alphabet, see Fig.2.

The relevance of this language is the use of two main concepts: accumulations (levels) and rate of change of accumulations (flow rates) to represent feedback structures. These two concepts have been well known for a long time in control theory as state variables and as its rate of change, respectively. The novel contribution of Professor Forrester [2] was to put them along with an experimentation process discipline with simulators in a way that yielded an approach with practical applications to a wide range of disciplines.

In this work, Level is used instead of Stock. Levels and Flows are the important concepts. They are the nouns and verbs in a language for representing movement. They are the two primary building blocks for representing the structure of any dynamic system.

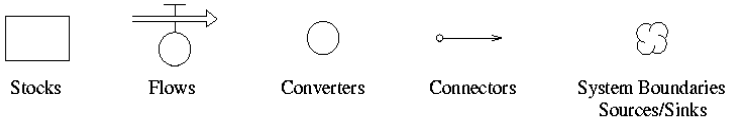


Fig. 2. Symbols from the *ithink* software

Level means level of accumulation of something. Accumulation of food in our stomach, money in a bank, knowledge in a team, love in our heart. At any time, accumulations indicate how things are going. They indicate the state of the system. If our stomach is too full, we are uncomfortable. If our motivation is high, we get good performance. The reason that accumulations matter to us is that it enables and inhibits actions. It acts both as resources and constraints. As resources, it enables actions. We can continue, for example, reading this paragraph because of our energy and knowledge reserves in our body. We would do nothing without accumulations.

It is important to denote two kinds of resources: Consumables and Catalysts. Energy, money, gasoline, and water are instances of consumables. Knowledge, love, self-confidence- and commitment are instances of catalysts. The difference is that during the action, the formers are consumed and the latter may not.

As constraints, the resources block the action. Accumulation of smoke in a room makes it difficult to breathe. Accumulation of frustration often inhibits effective work. In any case, levels of accumulation affect flows or activities.

Flows mean rates of change. Levels and flows always come together. There is no one without the other. If some accumulation of something appears then this event was the result of some flow of something, and vice versa. Activities leaves “tracks”, which in turn, stimulate further activities.

Converters are signified by circle. They are multi input - one output that represents static linear and nonlinear mapping. Converters are used to elaborate the details of the Flow rate and Level structure in the models. Converters can be used to provide an alternative measure for stock or as a “Roll-Up” of flows or as external inputs to the system, such as time series, step, ramp or impulse functions, among others.

Connectors reflect the idea “what depends on what”'. They represent the relationship among the elements of the system. They do not take numerical values. They just transmit them. This is the difference with flows.

Sources and Sinks indicate the boundaries of the model. They are signified by clouds and represent infinite sources, it does not matter to us what is in the clouds.

5 Case of Study

The approach was applied to elicit and represent the knowledge of the dynamic of a five star hotel by request of a consulting firm. The objective was to construct a simulator that allowed to shape discussions and to expose skills to make decisions among managers.

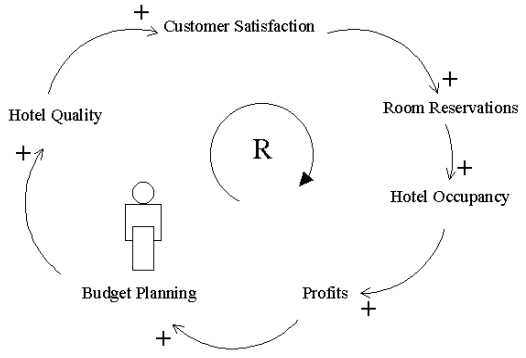


Fig. 3. Central loop of the Hotel

A team of three persons was integrated to realize the part of knowledge acquisition: a hotel administrator, a consultant of quality programs and a knowledge engineer. The target information from several narrative sessions was the central process of the business represented by one single loop. This central loop is shown in Fig. 3 and represents the circular reasoning about the nature of a hotel business.

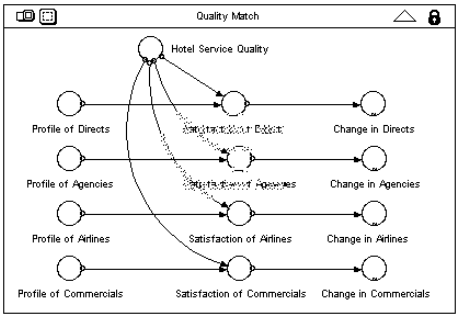


Fig. 4. Quality Match subsystem

The story says: (1) The bigger the customer satisfaction, the larger the number of room reservations; (2) the larger the number of reservations, the larger the number of customers in the future; (3) The larger number of customer, the bigger the hotel profits

(a strategy by volume was selected); (4) the bigger the profits, the bigger the budget to improve quality offered by the hotel, and (5) the better the hotel quality service, the bigger the customer satisfaction. Of course, these causal relationships also could have been written as if-then rules. This loop can also be considered as the myth that explains to the participants how the hotel business works. It is a reinforcement loop that implies an unstable system. This kind of systems presents exponential growing behavior in their dynamics.

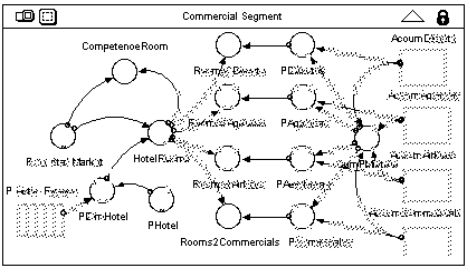


Fig. 5. Commercial Segment Subsystem

The next step was to obtain the differentiation of the elements. Four types of customers were declared: by walking (Directs), by travel agencies (Agencies), by airlines (Airlines), and by commercial arrangements (Commercials). Each customer with its own quality demand profile. The rest of the elements were also differentiated. Policies for budget planning were not represented by request of the consultants. These rules that close the loop will be contained only in the heads of the participants.

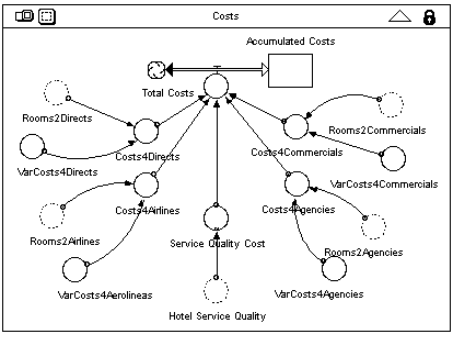


Fig. 6. Cost Subsystem

The participant will be able to observe customer satisfaction and profits and change the quality of the hotel prices as control variable to drive the business.

After differentiation, the next step is to construct hierarchical levels. Five subsystems were structured: (1) Quality Match, (2) Commercial Segments, (3) Costs, (4) Incomes, and (5) Customer Satisfaction Accumulation.

The subsystems were implemented with the *ithink* software. Quality Match is shown in Fig. 4 and represents the difference between the service quality offered by

the hotel and the one demanded by the customer. The difference provokes a change in the buying incentive of the customer. Commercial Segments, in Fig. 5, represents the number of rooms gained by our hotel against the competence. Costs, in Fig. 6, represents the fixed and variable costs. Incomes, in Fig. 7, represents the flow of money come from the four classes of customers. A delay of the two months is included in Agencies.

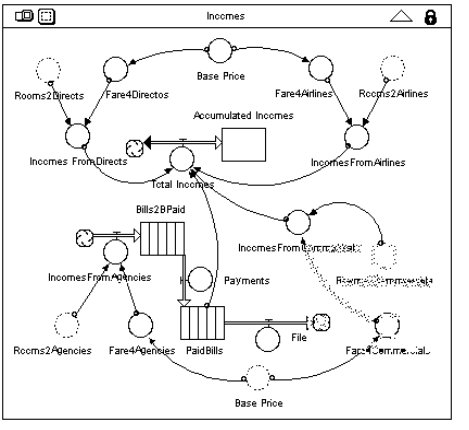


Fig. 7. Income Subsystem

The accumulators of Costs and Incomes are just used as arithmetic accumulators. Customer Satisfaction Accumulation, in Fig. 8, represents the core of the simulation model.

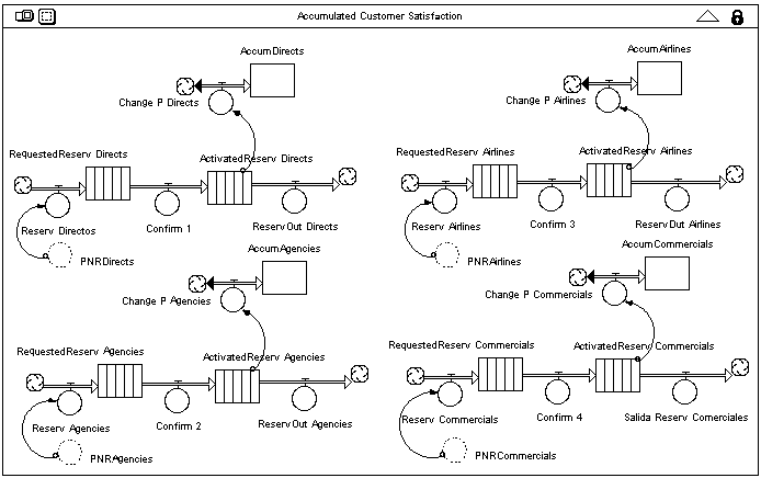


Fig. 8. Accumulated Customer Satisfaction Subsystem

The basic dynamic simulation model is formed by four first order systems with delay of two time units (months) and shown in Fig. 8. Room Reservations are converted in Active Reservations and it is then inferred the change in the accumulated customer satisfaction. The four systems, corresponding to each of the customer profiles, are distinguishable by the four containers. These systems are not independents since they share the number of customers according to the current accumulated customer satisfaction.

```
Accumulated Customer Satisfaction
AccumAgencies(t) = AccumAgencies(t - dt) + (Change_P_Agencies) * dt
  INIT AccumAgencies = .2
  INFLOWS:
    Change_P_Agencies = ActivatedReserv_Agencies-1
AccumAirlines(t) = AccumAirlines(t - dt) + (Change_P_Airlines) * dt
  INIT AccumAirlines = .35
  INFLOWS:
    Change_P_Airlines = ActivatedReserv_Airlines-1
AccumCommercials(t) = AccumCommercials(t - dt) + (Change_P_Commercials) * dt
  INIT AccumCommercials = .15
  INFLOWS:
    Change_P_Commercials = ActivatedReserv_Commercials-1
AccumDirects(t) = AccumDirects(t - dt) + (Change_P_Directs) * dt
  INIT AccumDirects = .3
  INFLOWS:
    Change_P_Directs = ActivatedReserv_Directs-1
ActivatedReserv_Agencies(t) = ActivatedReserv_Agencies(t - dt) + (Confirm_2 -
ReservOut_Agencies) * dt
  INIT ActivatedReserv_Agencies = 1
    TRANSIT TIME = 1
    INFLOW LIMIT = INF
    CAPACITY = INF
  INFLOWS:
    Confirm_2 = CONVEYOR OUTFLOW
  OUTFLOWS:
    ReservOut_Agencies = CONVEYOR OUTFLOW
```

Fig. 9. Segment of the generated numerical equations

The numerical equations are generated automatically by the ithink software. A segment of the output view is shown in Fig. 9.

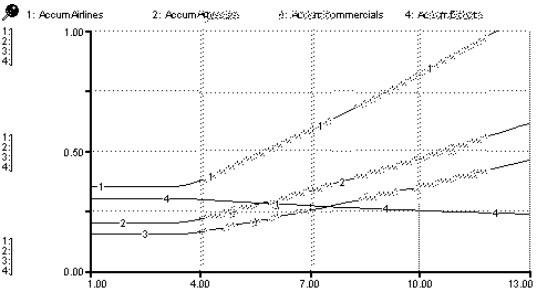


Fig. 10. Satisfaction of clients over one year given quality degree 4

The model represents the hypothesis that all of the customers are satisfied and the best profits are obtained at the end of the year by offering the highest quality (degree 5). Results by offering quality degree 4 are shown in Figures 10 and 11, all customers are satisfied except Directs, however positive profits are obtained after month 6

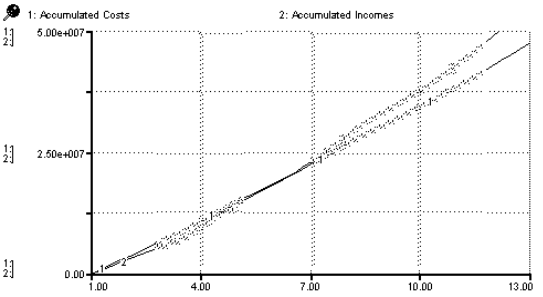


Fig. 11. Accumulated incomes and costs over one year given quality degree 4

Results by offering quality degree 5 are shown in Figures 12 and 13. All customers are satisfied and it should be expected that more profits might be obtained. However, the return of investment is slower than the previous case. Positive profit is obtained after month 8. Although, there were more incomes, also there were more expenses.

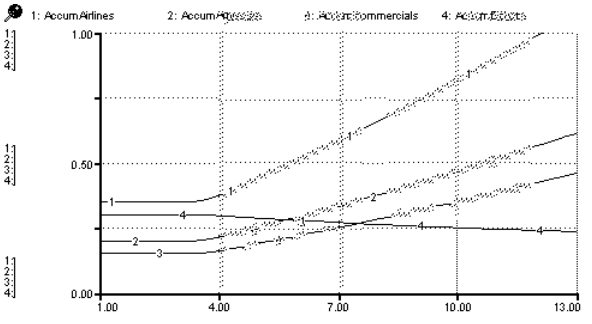


Fig. 12. Satisfaction of clients over one year given quality degree 5

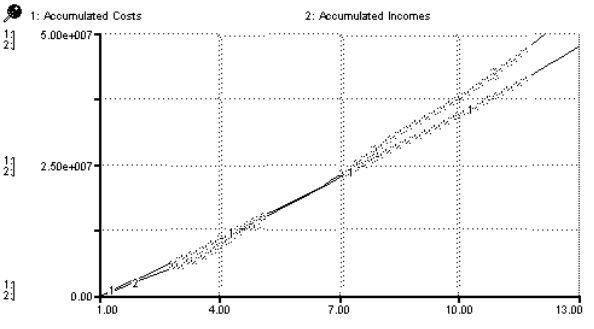


Fig. 13. Accumulated incomes and costs over one year given quality degree 5

This kind of counterintuitive idea is part of those who help to shape discussions in learning sessions among policymakers. So, knowledge and experiences can be transmitted and administrated in global organizations via simulators and experience sessions.

6 Conclusions

It has presented a case of qualitative knowledge acquisition as a process of Modeling for Learning Organizations. The approach is based on System Dynamics method and it has been presented as a general procedure to transmit knowledge as in old and modern cultures: myths, symbols, and numbers. A practical case about modeling of a five star hotel system illustrated the approach.

The paper addressed the first step out of three of a method for intelligence organizations. First step is to collect and represent qualitative knowledge into a computer simulation model. Second is to analyze past and new generated knowledge and third is to disseminate relevant data, information and experience to the organization. This method leads us to explore and look for answers on how to manage experiences.

One of the products obtained by the knowledge acquisition step is a system of equations in the form of state space representation commonly used in control theory. This system of equations allows observing the dynamic behavior of policymakers' mental models via simulation without the necessity of solving the equations. This system can be part of a Knowledge Based Simulation (KBS) with fuzzy inference to expand the capacity of Knowledge-Based Systems and try to answer questions of the type: Can situation A happen? What conditions must be present to observe event B? These systems overpass their predecessors by answering questions beyond "What if". In the same manner, Model Base Reasoning (MBR) techniques can expand their range of applications and not only to tackle problems based on scientific knowledge, but also applications in business and social areas where systems are considered soft. Designing this kind of systems has still a great component of intuition and good will. As it is known, the big challenge in our days is how to balance the development of our social systems with the development of our technological systems.

Intensive learning by modeling was observed in the participant team. Results showed satisfactory representation of qualitative knowledge into a computer simulation model. It can be said: having as a final result a set of to mathematical equations relationship such as customer satisfaction and profit.

Acknowledgement to José Luis a la Torre, coauthor of the *ithink* model and simulations and Armando Pérez, both partners of the modeling and learning experience. Also, for partial financial support to Prof. Rosa María Sánchez of DynExel-ITESM program; Prof. Carlos Narváez, Director of the Department of Control Engineering, and Prof. Francisco J. Cantú, Director of the Center for Artificial Intelligence at ITESM Campus Monterrey.

References

1. Davis, R.: Diagnostic Reasoning Based on Structure and Behavior. Artificial Intelligence. Vol. 24. No. 1-3, (1984) 347—410
2. Forrester, Jay W.: Collected Papers of Jay W. Forrester. Portland, Ore: Productivity Press, USA (1975)

3. Forrester, Jay W.: System Dynamics as a Foundation for Pre-College Education. In Anderson, D.F., Richardson, G.P., Sterman, J.D., (eds.): Volume I. System Dynamics '90, Mass: System Dynamics Society, USA (1990) 367—380
4. Gonzalez, A.J., Dankel, D.D.: The Engineering of Knowledge-Based Systems. Theory and Practice. Prentice Hall, USA (1993)
5. HPS96: An Introduction to Systems Thinking. ithink Analyst Version. High Performance Systems, Inc. USA (1996)
6. Kuipers, B.: Qualitative Simulation. Artificial Intelligence. Vol. 29. No. 3. (1986) 289—338
7. Kosko, B.: Neural Networks and Fuzzy Systems. A Dynamical Systems Approach to Machine Intelligence. Prentice Hall, Inc. USA (1992)
8. Lee, K.Ch., Lee, W.J., Kwon, O.B., Han, J.H., Yu, P.I.: Strategic Planning Simulation Based on Fuzzy Cognitive Map Knowledge and Differential Game. Simulation. Vol. 71. USA (1998)
9. Ljung, L., Glad, T.: Modeling of Dynamic Systems. Prentice Hall, USA (1994)
10. Nissen, M.E.: Designing Qualitative Simulation Systems for Business. Simulation & Gaming. Thousand Oaks. USA (1996)
11. Perusich, K.: Fuzzy Cognitive Maps for Policy Analysis. International Symposium on Technology and Society Technical Expertise and Public Decisions. (1996) 369—373
12. Rothenberg, J.: Artificial Intelligence and Simulation. In Balci, O., Sadowski R.P., Nance, R.E. (eds.): Proceedings of the 1990 Winter Simulation Conference, USA (1990)
13. Scarl, E.A., Jamieson,J.R., Delaune, C.I.: Diagnosis and Sensor Validation through Knowledge of Strucutre and Function. IEEE Transactions on Systems, Man and Cybernetics. Vol. SMC-17. No. 3. (1987) 360—368
14. Scarl, E. A.: Tutorial on Model Based Reasoning. Florida Artificial Intelligence Research Symposium, St. Petersburg, Fl:FLAIRS (1991)
15. Senge, P.M.: The Fifth Discipline. Doubleday New York (1990)
16. Shaker, S.M., Gembicki, M.P.: The War Room Guide to Competitive Intelligence. McGraw-Hill, USA (1999)
17. Sterman, J.D.,: Prologue. In : Morecroft, J.D.W., Sterman, J.D. (eds): Modeling for Learning Organizations. System Dynamics Series. Productivity Press EUA (1994)
18. Weld, S., De Kleer, J.: Readings in Qualitative Reasoning about Physical Systems. Morgan Kaufmann, USA (1990)
19. Widman, L.E., Loparo, K.A., Nielsen, N.R.: Artificial Intelligence, Simulation and Modeling. John Wiley & Sons, Inc. USA (1989)

Applying One-Sided Selection to Unbalanced Datasets

Gustavo E.A.P.A. Batista[1], Andre C.P.L.F. Carvalho[2], and Maria Carolina Monard[3]

[1] Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo. Silicon Graphics Brasil.
[2] Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo.
[3] Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo/ILTC.
Av. Carlos Botelho, 1465. Caixa Postal 668.
CEP 13560-970. São Carlos - SP. Brasil.
Phone: +55-16-273-9692. FAX: +55-16-273-9751.
{gbatista, andre, mcmonard}@icmc.sc.usp.br

Abstract. Several aspects may influence the performance achieved by a classifier created by a Machine Learning system. One of these aspects is related to the difference between the number of examples belonging to each class. When the difference is large, the learning system may have difficulties to learn the concept related to the minority class. In this work, we discuss some methods to decrease the number of examples belonging to the majority class, in order to improve the performance of the minority class. We also propose the use of the VDM metric in order to improve the performance of the classification techniques. Experimental application in a real world dataset confirms the efficiency of the proposed methods.

1 Introduction

Supervised learning is the process of automatically creating a classification model from a set of instances, called the *training set*, which belong to a set of classes. Once a model is created, it can be used to automatically predict the class of other unclassified instance.

In other words, in supervised learning, a set of n training examples is given to an inducer. Each example $\mathbf{X}$ is an element of the set $F_1 \times F_2 \times \ldots \times F_m$ where F_j is the domain of the jth feature. Training examples are tuples $(\mathbf{X}, Y)$ where Y is the label, output or class. The Y values are typically drawn from a discrete set of classes $\{1, \ldots, K\}$ in the case of *classification* or from the real values in the case of *regression*. Given a set of training examples, the learning algorithm (*inducer*) outputs a *classifier* such that, given a new instance, it accurately predicts the label Y.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 315–325, 2000.

In this work we refer to classification and, for simplicity, we consider two class problems. However, the methods here described can easily be adapted and applied to problems with more than two classes.

One of the problems in supervised learning is learning from unbalanced training sets. For a number of application domains, for instance the diagnostic of rare diseases, a huge disproportion in the number of cases belonging to each class is common. In these situations, the design of a high precision classifier is straightforward: just classifying every new case as belonging to the majority class. However, accurate classification of minority class cases is frequently the major objective of such applications.

Many traditional learning systems are not prepared to induce a classifier that accurately classify both classes under such situation. Frequently the classifier has a good classification accuracy for the majority class, but its accuracy for the minority class is unacceptable.

In this work we discuss several methods for selecting training cases labelled with the majority class in order to improve the classification accuracy of the minority class. The main idea is to select and remove cases from the majority class while leaving untouched cases from the minority class. However, only cases that have low influence on learning the majority class concept are removed, such that classification accuracy of this class is not deeply affected. Removing cases from the majority class tends to decrease the disproportion between the classes, and as a result, to improve the minority class classification. Selection methods that focus on decreasing the number of cases from the majority class are known as *one-sided selection methods* [8].

In this work we propose some improvements to the one-sided selection methods proposed in [8] through the use of a powerful method to measure distance of symbolic attributes called *Value Difference Metric (VDM)*. Kubat's method is based on instanced-based learning systems. This sort of learning systems typically store the training set and use a distance metric to classify new cases. An Euclidean distance can be used to compare examples with continuous attributes values. However, whenever the dataset has symbolic attributes, instance-based systems frequently use very simple metrics, such as overlap metric (same attribute values have distance equal to zero while different attribute values have distance equal to one). Metrics like overlap metric may fail to capture the complexity of the domain, causing an inadequate performance of one-sided selection methods.

This work is organised as follows: Section 2 gives a general idea of why several learning systems are not prepared to cope with an unbalanced dataset. Section 3 explains why error rate and accuracy are not good metrics to measure the performance of learning systems trained on unbalanced datasets, and describes other metrics to substitute them. Section 4 describes the one-sided selection methods and Section 5 briefly describes the VDM metric for symbolic attributes. Some initial experiments using the MineSet tool from SGI are described in section 6, and performance measures are show in Section 7 in order to verify the effectiveness of the proposed methods. Finally, Section 8 shows the conclusions of this work.

2 Why Unbalanced Datasets Harm?

Learning from unbalanced datasets is a difficult task since most learning systems are not prepared to cope with a large difference between the number of cases belonging to each class. However, real world problems with these characteristics are common. Researchers have reported difficulties to learn from unbalanced datasets in several domains, such as retrieving information from texts [10], diagnosing rare diseases [3], detecting credit card fraud operations [13], and others.

Why learn under such conditions is so difficult? Imagine the situation illustrated in Figure 1, where there is a large unbalance between the majority class (-) and the minority class (+). It also shows that there are some cases belonging to the majority class incorrectly labelled (noise). Spare cases from the minority class may confuse a classifier like *k-Nearest Neighbour (k-NN)*. For instance, 1-NN may incorrectly classify many cases from the minority class (+) because the nearest neighbour of these cases are noisy cases belonging to the majority class. In a situation where the unbalance is very high, the probability of the nearest neighbour of a minority class case (+) be a case of the majority class (-) is near 1, and the minority class error rate will tend to 100%, which is unacceptable for many applications.

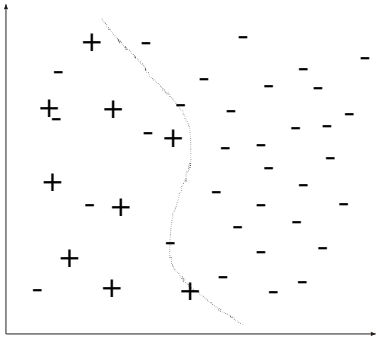


Fig. 1. Many negative cases against some spare positive cases.

Decision trees also suffer from a similar problem. In the presence of noise, decision trees may become too specialised (overfitting), i.e., the decision tree inducer may need to create many tests to distinguish the minority class cases (+) from noisy majority class cases. Pruning the decision tree does not necessarily alleviate the problem. This is due to the fact that pruning remove some branches considered too specialised, labelling new leaf nodes with the dominant class on that node. Thus, there is a high probability that the majority class will also be the dominant class of those leaf nodes.

Theoretical analysis show that Multi-Layer Perceptron networks (MLP) approximate a posterior bayesian probability which is independent of prior bayesian probability. However, empirical analysis have shown that MLP neural networks

have difficulty to learn when trained with unbalanced datasets [1]. According to [9], such disagreement between empirical and theoretical analysis is due to some suppositions made in theoretical analysis, such as a infinite training dataset, reaching the global minimum while training, and others.

3 Metrics to Measure Performance with Unbalanced Datasets

The error rate (E) and the accuracy (1-E) are widely used metrics for measuring the performance of learning systems. However, when the prior probability of the classes is very different, such metrics might be misleading. For instance, it is straightforward to create a classifier having 90% accuracy if the dataset has a majority class with 90% of the total number of cases, by simply labelling every new case as belonging to the majority class. Other factor against the use of accuracy (or error rate) is that these metrics consider different classification errors as equally important. For instance, a sick patience diagnosed as healthy might be a fatal error while a healthy patience diagnosed as sick is considered a much less serious error since this mistake can be corrected in future exams. On domains where misclassification cost is relevant, a cost matrix could be used. A cost matrix defines the misclassification cost, i.e. a penalty for making a mistake for each different type of error. In this case, the goal of the classifier is to minimise classification cost instead of error rate.

Different types of errors and hits performed by a classifier can be summarised as a *confusion matrix*. Table 1 illustrates a confusion matrix for two classes.

Table 1. Different types of errors and hits for a two classes problem.

	Positive Prediction	*Negative Prediction*
Positive Class	True Positive (a)	False Negative (b)
Negative Class	False Positive (c)	True Negative (d)

From such matrix it is possible to extract a number of metrics to measure the performance of learning systems, such as error rate $\frac{(c+b)}{(a+b+c+d)}$ and accuracy $\frac{(a+d)}{(a+b+c+d)}$. Other two metrics directly measure the classification error rate on the positive and negative class:

- False negative rate: $\frac{b}{a+b}$ is the percentage of positive cases misclassified as belonging to the negative class;
- False positive rate: $\frac{c}{c+d}$ is the percentage of negative cases misclassified as belonging to the positive class.

Figure 2 shows a common relationship among error rate, false positive rate and false negative rate. This analysis aimed to identify fraudulent credit card transactions (minority class, represented as negative class) into valid transactions

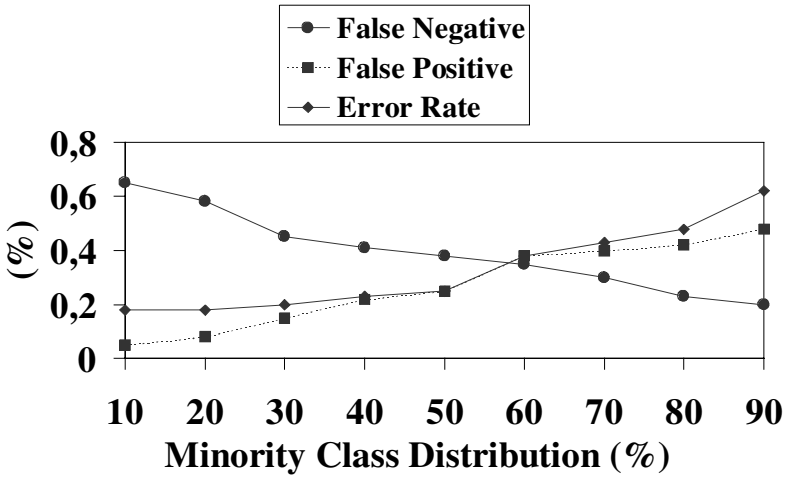


Fig. 2. Error rates with different class distributions in the training class.

(majority class, represented as positive class). Chan and Stolfo trained the C4.5 learning system [11] with different class distributions in the training class (the test class distribution was kept untouched) [4]. The chart begins with a training set consisting of 90% of cases from the majority class. The proportion of minority class cases is increased by 10% at each iteration. This increase in the number of minority class cases in the training set causes an improvement in the classification of cases from this class. However, the classification accuracy for the majority class decreased. The error rate on the test set, on the other hand, increased influenced by the bad performance of the majority class, which dominates the test set.

This experiment showed a decrease in the majority class accuracy as more cases from the minority class were added to the training set. In real situations, when it is difficult to find more cases of the minority class, a more suitable solution is to remove majority class cases, in order to get a more balanced distribution among the classes. However, a question stands: is it possible to reduce the number of the majority class cases without loosing too much performance on the majority class? This is the main objective of one-sided selection.

4 One-Sided Selection

The problem of unbalanced datasets has been studied and some solutions have been proposed. One of them, one-sided selection [8] proposes a careful removal of cases belonging to the majority class while leaving untouched all cases from the minority class, because such cases are too rare to be lost, even though some of them might be noise. Such careful removal consists of detecting and removing cases considered less reliable, using some heuristics. These heuristics can be better understood by dividing the cases into four distinct groups:

1. Mislabelled cases (noise). For instance, the majority class cases (-) located in the left region of Figure 1;
2. Redundant cases. Such cases might be represented by other cases that are already in the training set. For instance, the cases that are far from the decision border, like those located on the right top region of Figure 1;
3. Cases close to the decision border (borderlines). These cases are quite unreliable since even a small quantity of noise can move them to the wrong side of the decision border;
4. Safe cases. Those cases that are neither too close to the decision border nor are too far from it. These cases should be kept for learning.

The one-sided selection technique aims to create a training set consisting of safe cases. In order to achieve that, noisy, borderline and redundant majority class cases should be eliminated.

Borderline and noisy cases can be detected by *Tomek links* [14]. Given the cases x and y belonging to different classes and be $d(x, y)$ the distance between x and y. A (x, y) pair is called a Tomek link if there is not a case z, such that $d(x, z) < d(x, y)$ or $d(y, z) < d(y, x)$. Cases that are Tomek links are borderline or noise. Figure 3 illustrates a dataset obtained by the removal of the majority class cases (see Figure 1) that formed Tomek links.

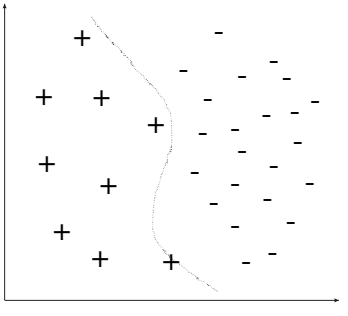


Fig. 3. Dataset without noisy and borderline cases.

Redundant cases can be deleted by finding a *consistent subset*. By definition, a consistent subset $C \subset S$ is consistent with S if, using a 1-nearest neighbour (1-NN), C correctly classify the cases in S [6]. An algorithm to create a subset C from S is the following: First, randomly draw one majority class case and all cases from the minority class and put these cases in C. Afterwards, use a 1-NN over the cases in C to classify the cases in S. Every misclassified case from S is moved to C. It is important to note that this procedure does not find the smallest consistent subset from S. Figure 4 shows the dataset (see Figure 1) after the removal of redundant cases from the majority class.

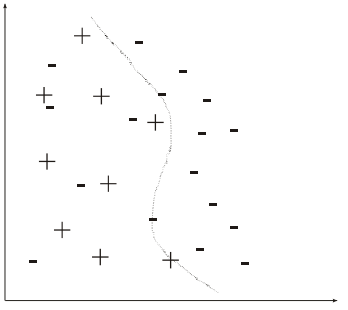


Fig. 4. Dataset without redundant cases.

5 The VDM Metric

In [12] a powerful method to measure distance of symbolic attributes called *Value Difference Metric (VDM)* is presented. In contrast to simpler methods, for instance the overlap metric, which measure the distance between symbolic attributes by just verifying if the attribute has the same value (distance equals to 0) or not (distance equals to 1). The VDM metric considers the classification similarity for each possible value of an attribute to calculate the distances between these values. As a result, a matrix of distances is created from the training set for each attribute. The distance d between two values for a certain symbolic attribute V is defined by:

$$d(V_1, V_2) = \sum_{i=1}^{n} \mid \frac{C_{1_i}}{C_1} - \frac{C_{2_i}}{C_2} \mid^k$$

In this equation, V_1 and V_2 are two values assumed by the attribute V. n is the number of classes in the dataset and C_{1_i} is the number of cases belonging to the class i whose attribute V assumed the value V_1. C_1 is the total number of cases whose attribute V assumed the value V_1 and k is a constant, frequently 1.

The VDM metric considers two values to be similar if they occur with almost identical relative frequencies for all classes. The ratio $\frac{C_{1_i}}{C_1}$ represents the probability of a case to be classified as belonging to the class i given that the attribute V has the value V_1.

The VDM metric, used in this work, has the following characteristics:

1. $d(a, b) > 0, a \neq b$;
2. $d(a, b) = d(b, a)$;
3. $d(a, a) = 0$;
4. $d(a, b) + d(b, c) \geq d(a, c)$.

6 Initial Experiments

An initial investigation about the behaviour of one-sided selection methods was carried out using the *Breast Cancer* dataset from the UCI repository [3]. This

dataset allows to visualise in three dimensions two cluster of cases that represent the classes *cancer=malignant* and *cancer=benign*. The following attributes were used as axes in the visualisation: *clump thickness, uniformity of cell size* and *bare nuclei*. Figure 5 shows the original dataset visualised through the Mineset tool from SGI.

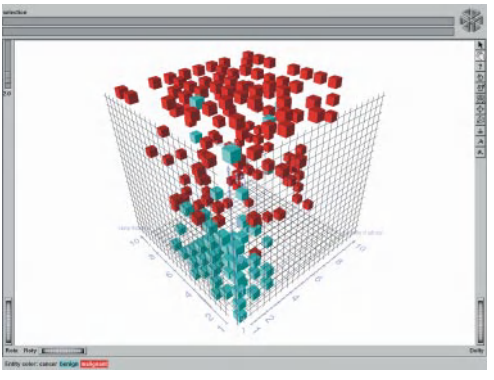


Fig. 5. Original Breast Cancer dataset.

First of all, cases belonging to the *cancer=benign* class (represented by light colour) that formed Tomek links were removed, since this class has some sparely distributed cases that might be noise. Afterwards, a consistent subset of the cases belonging to the *cancer=malignant* class (represented by dark colour) was found, since this class have some cases far from the border. Figure 6 shows the dataset after the application of one-sided selection methods.

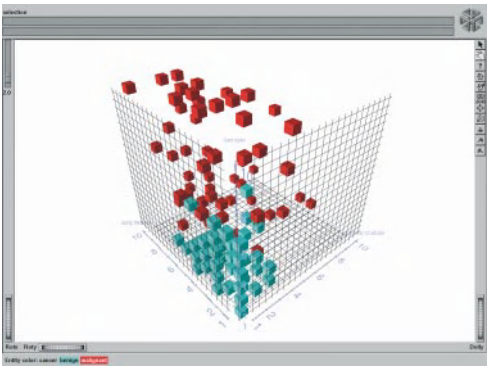


Fig. 6. Breast Cancer dataset without redundant, borderline and noisy cases.

7 Experimental Results

Some experiments were conducted to verify whether one-sided selection is able to improve the classification of the minority class in unbalanced datasets. The C5.0 learning system and the *Hepatitis* dataset from UCI [3] were used for these experiments. The Hepatitis dataset has 155 cases with 123 cases (79,3%) belonging to the class *live* and 32 cases (20,6%) belonging to the class *die*. Also the Hepatitis dataset has both numeric and symbolic attributes whose distance was computed using the VDM metric. These experiments were performed with the aid of the AMPSAM environment [2] to facilitate performance measures.

The hepatitis dataset is know by the Machine Learning community for its difficulty to produce good performance results. According to [7], very few learning systems obtained an accuracy higher than two percent points over the *baseline accuracy*, i.e., an accuracy two percent points over 79,3%.

C5.0 performance was measured on the original dataset with all cases (a); on the dataset without noisy and borderline cases removed by Tomek links (b); on the dataset without redundant cases removed by creating a consistent subset (c); on the dataset without noisy, borderline and redundant cases (d); and, finally, on the dataset without some majority class cases removed randomly (e). The error rates were measured using the 3-fold cross validation resampling method. The number of partitions $k = 3$ was chosen because there are very few minority class cases and higher k values might lead to high variance results. To confirm these results, 3-fold cross validation was applied 3 times. Since the results were similar, only the results of one of these experiments is shown in Table 2. The column N shows the number of examples in the training set (majority/minority class). Fn and Fp are the false negative and false positive rates, respectively. Columns $\delta(Fn)$ and $\delta(Fp)$ are the standard deviation for the false negative and false positive rates, respectively.

Table 2. Error rates and standard deviation.

	N	Fn	$\sigma(Fn)$	Fp	$\sigma(Fp)$
a	80/23	10%	5,8%	62%	5,1%
b	70/23	15%	4,0%	43%	11,7%
c	62/23	13%	8,6%	57%	11,7%
d	55/23	28%	4,5%	29%	7,9%
e	50/23	9%	2,6%	57%	17,4%

Comparing cases b and a using the standard hypothesis testing model the difference is significant for the minority class although not degrading significantly for the majority class. Comparing cases d and a as well as d and e there is a significant improvement for the minority class but there is also a significant degradation for the majority class.

The results obtained suggest that one-sided selection can effectively decrease the error rate of the class *die* (minority class), specially when Tomek links are used (*b* and *d*). However, the random selection method (which does not use any heuristic) obtained some results comparable to the selection through consistent subsets method. Even though random selection does not apply any heuristic, it has the merit of removing the cases with equal probability. Probably, random selection is the method that causes the smallest change in the dataset distribution, comparing with the methods used in this work. There may exist other reasons for this effect, such as: most of the UCI datasets have already been carefully analysed before being made available in the repository. During these analysis, many borderline, noisy and redundant cases might have been removed, decreasing the effectiveness of the heuristics considered in this work. Another reason could be that minority class cases were left untouched, even though there might be noisy minority class cases. This decision was taken due to the necessity of keeping all the few cases belonging to the minority class. However, noise present in the minority class cases can reduce the classification accuracy. In this work, only majority class cases forming Tomek links were removed. However, Tomek links might be related to noisy cases from the minority class. By only removing majority class cases forming Tomek links, minority class noisy cases may not be removed. Moreover, important majority class cases may be removed. Since minority class cases are precious, an improvement to the current method is to distinguish between noisy and borderline cases. Unfortunately, Tomek links do not offer a safe method to distinguish such types of cases and other methods should be investigated.

8 Conclusions

Datasets with a large difference between the number of cases belonging to each class are common in Machine Learning. For those datasets, it may be trivial to produce a classifier that performs very well on the majority class cases. However, accurately classify the cases of the minority class is much more difficult to achieve. Several practitioners have been solving this problem by replicating (sometimes with little noise) minority class cases. However, this technique potentially introduces bias in the dataset. One-sided selection does a careful removal of majority class cases, aiming to balance the number of cases of each class. This work shows some results that seem to suggest the effectiveness of this technique which is actually being implemented as part of a computational system for data pre-processing. In future works, we will look for new heuristics to discriminate between borderline and noisy cases in order to select and remove noisy minority class cases. Also, we will look for new heuristics to select majority class cases.

Acknowledgements. The authors wish to thank an anonymous referee of MICAI-2000 for his/her helpful comments on this paper. This research was partly supported by FINEP and Silicon Graphics Brasil.

References

1. Barnard, E.; Cole, R.A.; Hou, L. *Location and Classification of Plosive Constants Using Expert Knowledge and Neural Nets Classifiers.* Journal of the Acoustical Society of America, 1988, 84 Supp 1:S60.
2. Batista, G.E.A.P.A.; Monard, M.C. *A Computational Environment to Measure Machine Learning Systems Performance* (in Portuguese). Proceedings I ENIA, 1997, p. 41-45.
3. Blake, C.; Keogh, E.; Merz, C.J. *UCI Repository of Machine Learning Databases.* Irvine, CA: University of California, Department of Information and Computer Science. http://www.ics.uci.edu/ mlearn/MLRepository.html.
4. Chan, P.K.; Stolfo, S.J. *Learning with Non-uniform Class and Cost Distributions: Effects and a Distributed Multi-Classifier Approach.* KDD-98 Workshop on Distributed Data Mining, 1998, p 1-9.
5. Cost, S.; Salzberg, S. *A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features.* Machine Learning, 10 (1), 1993, pp. 57-78.
6. Hart, P.E. *The Condensed Nearest Neighbor Rule.* IEEE Transactions on Information Theory, IT-14, 1968, pp. 515-516.
7. Holte, C.R. *Very Simple Classification Rules Perform Well on Most Commonly Used Datasets.* Machine Learning, 11, 1993, pp. 63-91.
8. Kubat, M.; Matwin, S. *Addressing the Course of Imbalanced Training Sets: One-Sided Selection.* Proceedings of the 14th International Conference on Machine Learning, ICML'97, Morgan Kaufmann, 1997, pp. 179-186.
9. Lawrence, S.; Burns, I.; Back, A.; Tsoi, A.C.; Giles, C.L. *Neural Network Classification and Prior Class Probabilities.* Tricks of the trade, Lecture Notes in Computer Science State-of-the-art surveys, G. Orr, K.R. Müller, R. Caruana (editors), Springer Verlag, 1998, pp. 299-314.
10. Lewis, D.; Catlett, J. *Heterogeneous Uncertainty Sampling for Supervised Learning.* Proceedings of the 11th International Conference on Machine Learning, ICML94, Morgan Kaufmann, 1994, pp. 148-156.
11. Quinlan, J.R. *C4.5 Programs for Machine Learning.* Morgan Kaufmann Publishers, CA, 1988.
12. Stanfill, C.; Waltz, D. *Toward Memory-Based Reasoning.* Communications of the ACM, 29(12), 1986, pp. 1213-1228.
13. Stolfo, S.J.; Fan, D.W.; Lee, W.; Prodromidis, A.L.; Chan, P.K. *Credit Card Fraud Detection Using Meta-Learning: Issues and Initial Results.* Proc. AAAI-97 Workshop on AI Methods in Fraud and Risk Management, 1997.
14. Tomek, I. *Two Modifications of CNN.* IEEE Transactions on Systems Man and Communications, SMC-6, 1976, pp. 769-772.

NSL/ASL: Distributed Simulation of Modular Neural Networks

Alfredo Weitzenfeld, Oscar Peguero, and Sebastián Gutiérrez

Departmento Académico de Computación
Instituto Tecnológico Autónomo de México (ITAM)
Río Hondo #1, San Angel Tizapán, CP 01000
México DF, MEXICO
email: alfredo@lamport.rhon.itam.mx

Abstract. As neural systems become large and complex, sophisticated tools are needed to support effective model development and efficient simulation processing. Initially, during model development, rich graphical interfaces linked to powerful programming languages and component libraries are the primary requirement. Later, during model simulation, processing efficiency is the primary concern. Workstations and personal computers are quite effective during model development, while parallel and distributed computation become necessary during simulation processing. We give first an overview of modeling and simulation in NSL together with a depth perception model example. We then discuss current and future work with the NSL/ASL system in the development and simulation of modular neural systems executed in a single computer or distributed computer network.

Keywords: Neural Networks, Simulation, Modular, Distributed

1 Introduction

The first task in modeling a neural system is creating a desired neural architecture. Creating a neural architecture involves choosing appropriate data representations for neural components while specifying their corresponding interconnections and full network dynamics. Additionally, network input and control parameters are selected. When modeling biological systems designed to reproduce anatomical and physiological data in a faithful way, there are many ways to characterize a neuron. The complexity of the neuron depends primarily on the accuracy needed. For example, compartment models [1] are quite successful in modeling detailed electric transmission in neurons. When behavioral analysis is desired, the neural network as a whole may often be adequately analyzed using simpler neuron models such as the analog leaky integrator model [2]. And sometimes even simpler neural models are enough, as the discrete McCulloch-Pitts binary model [3] where a neuron is either on or off during each time step. The last two neural models are quite appropriate when modeling analog or binary artificial networks, respectively.

A large number of simulation systems have been developed to support different types of neural modeling [4][5]. Among these, the most outstanding at the single

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 326—337, 2000.

neuron level are GENESIS [6] and NEURON [7] specifically designed to model cells taking into account their detailed morphology. For large-scale neural networks, simulators such as Aspirin/MIGRAINES [8] and NSL - Neural Simulation Language [9] have been developed to support both biological and artificial systems.

2 Neural Modeling

While neural modeling is carried out at many levels of granularity, we emphasize in NSL modularity at each of these levels. In such a way, a neural architecture is described at the highest level of abstraction in terms of multiple *neural modules* representing the overall neural functionality in the model specified each in terms of their underlying *neural networks* implementations. Thus, a complete neural architecture would consist of (1) a set of interconnected neural modules and (2) a set of interconnected neurons specifying each neural module.

2.1 Neural Modules

Neural modules in NSL are hierarchically organized in a tree structure where a root module may be further refined into additional *submodules* as shown in Figure 1. The hierarchical module decomposition results in *module assemblages* – a network of submodules seen in their entirety in terms of a single higher-level neural module. Modules (and submodules) may be implemented independently from each other in both a top-down and bottom-up fashion, an important benefit of modular design.

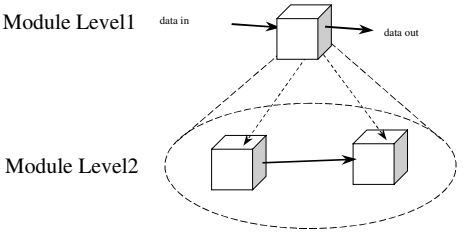


Fig. 1. NSL computational model is based on hierarchical interconnected modules. A module at a higher level (level 1) is decomposed (dashed lines) into two interconnected (solid arrow) submodules (level 2).

The actual module representation includes a set of unidirectional input and output *data ports* supporting external communication between modules. These *ports* represent module entry or exit points, used to receive or send data from or to other modules, as shown in Figure 2.

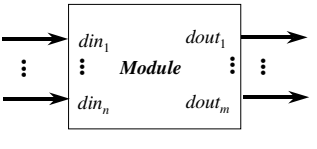


Fig. 2. Each Module may contain multiple input, $din_1,...,din_n$, and output, $dout_1,...,dout_m$, data ports for unidirectional communication.

2.2 Neural Networks

Each neural module is implemented by its underlying neural network and may recruit any number of neurons for its implementation, as shown in Figure 3.

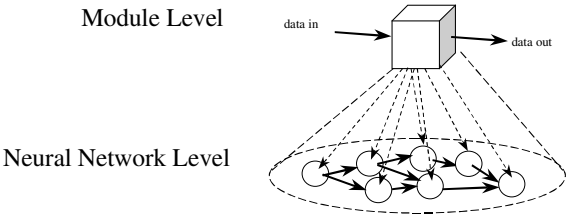


Fig. 3. A module encapsulates a neural network. Although neurons could be treated themselves as modules for further refinement, we treat them as separate entities, thus drawing them as spheres instead of cubes.

While different neural models have been simulated with NSL, we consider at the highest level a "simple" neuron having its internal state described by a single scalar quantity or membrane potential *mp*, input *s* and output *mf*, specified by some nonlinear function, as shown in Figure 4.

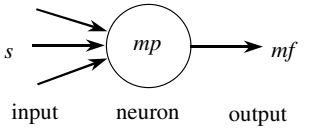


Fig. 4. Single compartment neural model represented by its membrane potential *mp*, and firing *mf*. *s* represents the set of inputs to the neuron.

The neuron may receive input from many different neurons, with only a single output branching to many other neurons. The choice of transformation from *s* to *mp* defines the particular neural model utilized. The *leaky integrator* model is described by

$$\tau \frac{dmp(t)}{dt} = -mp(t) + s(t)$$

where the *average firing rate* or output of the neuron, *mf*, is obtained by applying some "threshold function" to the neuron's membrane potential where σ is usually described by a non-linear function also known as a *threshold* function, such as *ramp*, *step*, *saturation* or *sigmoid*,

$$mf(t) = \sigma(mp(t))$$

The neural network itself is made of any number of interconnected neurons, where the most common formula for the input to a neuron is

$$sv_j = \sum_{i=0}^{n-1} w_i uf_i(t)$$

where $uf_i(t)$ represents the firing of neuron u_i whose output is connected to the *j*th input line of the neuron v_j, and w_i is the weight on that link (*up* and *vp* are analogous to *mp*, while *uf* and *vf* are analogous to *mf*).

2.3 Depth Perception Model

In depth perception, a three dimensional scene presented to the left eye differs from that presented to the right eye. A single point projection on each retina corresponds to a whole ray of points at different depths in space, but points on two retinae determine a single depth point in space, the intersection of the corresponding projectors, as shown in Figure 5.

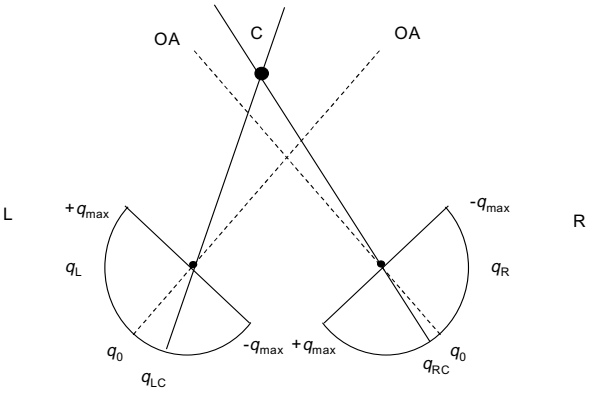


Fig. 5. Target C in space is projected to the two retinae L and R at different *disparity* points, q_{LC} and q_{RC}.

A particular problem in depth perception is that of ghost targets, where for example as shown in Figure 6, the projections for targets A and B correspond to the same projections from ghost targets C and D resulting in ambiguities.

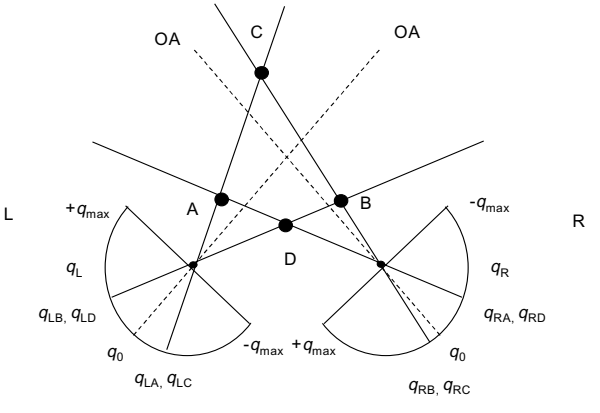


Fig. 6. Targets A and B generate the same projections as ghost targets C and D in the two retinae.

The depth perception model developed by House [10] uses two systems to build a depth map, one driven by *disparity* cues while the other is driven by *accommodation* cues. The accommodation driven field receives information about focal length and - left to its own devices - sharpens up that information to yield depth estimates. The disparity driven-field receives difference in retina projection to suppress ghost targets.

Both accommodation and disparity depth maps are described by a similar neural network as shown in Figure 7.

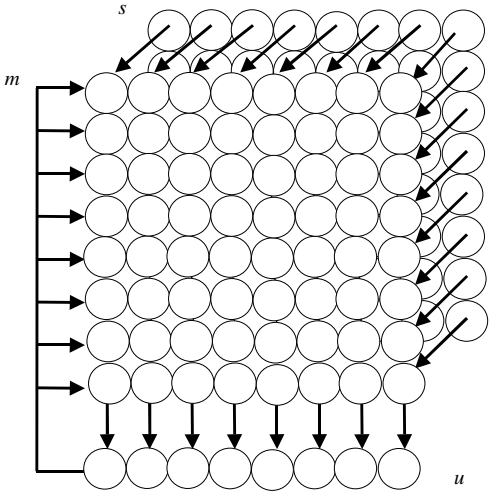


Fig. 7. Depth map neural network made of three layers *s*, *m* and *u*, where each layer is made of a set of homogeneous neurons spatially organized as shown.

The equations for the network are as follows:

$$\tau_m \frac{\partial m_{ij}}{\partial t} = -m_{ij} + w_m * f(m_{ij}) - w_u * g(u_j) - h_m + s_{ij}$$

$$\tau_u \frac{\partial u_j}{\partial t} = -u_j + \sum_i f(m_{ij}) - h_u$$

$$f(m_{ij}) = step(m_{ij}, k)$$

$$g(u_j) = ramp(u_j)$$

At the highest level each neural network corresponds to a single neural module as shown in Figure 8.

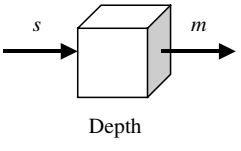


Fig. 8. Each depth map neural network is stored as a single neural module with input *s* and output *m*.

The two depth systems are intercoupled so that a point in the accommodation field excites the corresponding point in the disparity field, and viceversa. This intercoupling is shown at the neural network level in Figure 9.

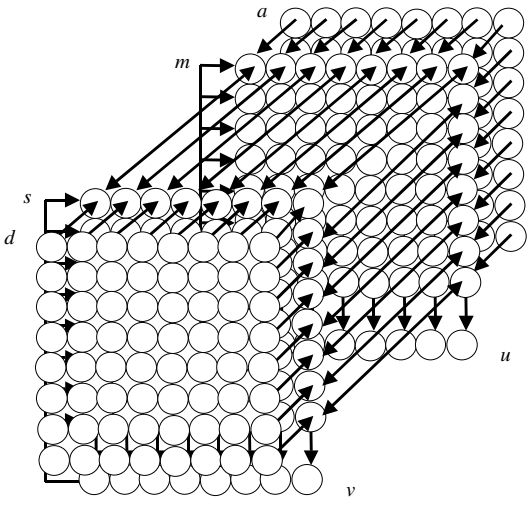


Fig. 9. Intercoupled accommodation and disparity depth maps made of layers *a*, *m* and *u*, and layers *d*, *s* and *v*, respectively.

The equations for the disparity depth map are as follows:

$$\tau_s \frac{\partial s_{ij}}{\partial t} = -s_{ij} + w_s * f\left(s_{ij}\right) + w_t * f\left(t_{ij}\right) - w_v * g\left(v_j\right) - h_s + d_{ij}$$

$$\tau_v \frac{\partial v_j}{\partial t} = -v_j + \sum_i f\left(s_{ij}\right) - h_v$$

$$f\left(s_{ij}\right) = sigma\left(s_{ij}\right)$$

$$g\left(v_j\right) = ramp\left(v_j\right)$$

The equations for the accommodation depth map are as follows:

$$\tau_m \frac{\partial m_{ij}}{\partial t} = -m_{ij} + w_m * f\left(m_{ij}\right) + w_t * f\left(t_{ij}\right) - w_u * g\left(u_j\right) - h_m + a_{ij}$$

$$\tau_u \frac{\partial u_j}{\partial t} = -u_j + \sum_i f\left(m_{ij}\right) - h_u$$

$$f\left(m_{ij}\right) = sigma\left(m_{ij}\right)$$

$$g\left(u_j\right) = ramp\left(u_j\right)$$

The corresponding NSL model consists of three interconnected modules: *Retina* *r*, *Depth* (accommodation) *m* and (disparity) *s*, as shown in Figure 10. (Note that *Retina* and *Depth* correspond to module types, while *r*, *m* and *s* represent module instantiations, where *m* and *s* have similar type definition).

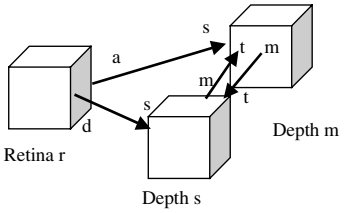


Fig.10. The *Retina* module contains two output ports, *dp* and *ap*, for disparity and accommodation, respectively. The *Depth* module consists of an input port *sp*, receiving data from the *Retina*, a second input port *tp*, receiving input from the other *Depth* module, and an output port *mp*.

3 Neural Simulation

Simulation of neural networks varies depending on whether it relates to artificial or biological systems. Artificial neural networks particularly those involving learning usually require a two-stage simulation process, (1) an initial training phase and (2) a subsequent processing or running phase. On the other hand, biological networks usually require a single running phase. The simulation process involves interactively specifying aspects of the model that tend to change, such as network input and parameter values, as well as simulation control and visualization. In general, the user analyzes output generated by the simulation, both visual and numerical, deciding if any modifications are necessary. If network input or parameter changes are necessary, these may be interactively specified, and the model is simulated again with the newly specified changes. On the other hand, if structural modifications in the neural architecture are required, then the developer must go back to the modeling phase.

In terms of execution, concurrency plays an important role in neural network simulation, not only as a way to increase processing performance but also to model neurons more faithfully [11]. Yet, most simulation systems execute sequentially due to lower complexity in their underlying hardware and software. On the other hand, distributed processing can be a more effective solution to parallel processing requiring the use of more sophisticated and expensive multiprocessor computers as opposed to a network of "standard" computers in the former case. The NSL computational model enables both sequential and distributed processing (as well as parallel processing) with very limited overhead from the user's side.

3.1 Sequential Simulation

Under sequential neural network simulation, neural dynamics are executed one equation at the time where the output from one neuron is immediately fed as input to the next one. In NSL, the same procedure is applied to modules where each module would get executed in a predetermined order sending newly generated output port values immediately to their corresponding interconnected input ports. This approach simplifies simulation results in that the order of computation depends directly from the

order of computation. To further simulate brain-like concurrent behavior while doing sequential processing, NSL interleaves module execution while buffering output from one module to the next one, making processing order unimportant. Yet, this approach does not improve processing efficiency.

3.2 Distributed Simulation

The *ASL - Abstract Schema Language* [12] is a distributed architecture integrating with the NSL simulation system to support both single processor and multiprocessor distributed computation. (The term *schema* [13] in ASL corresponds to *active* or concurrent objects [14] with the ability to process concurrently.) ASL extends the NSL module and port interconnection model adding *asynchronous* inter-process communication, especially suitable to optimize communication performance. In such an environment, all simulation control as well as graphical displays take place in the *console* in one of the machines responsible for distributing processing among local and/or remote machines. The actual distribution is specified by the user or automatically generated by the system depending on the available machines in the network. In ASL, module execution is managed by a *Thread Manager* in each process, while the *IPC* (*Inter-Process Communication*) *Manager* is responsible for external data communication. The current implementation of ASL is in C++ using *PVM* [15] as the IPC Manager and *PThreads* [16] as the Thread Manager.

3.3 Depth Perception Simulation

Under both sequential and distributed environments, simulation of the depth perception model generates similar results although with different efficiency. In Figure 11 we show the initial input to the model. Figure 12 shows two different targets with the corresponding initial accommodation (top right) and disparity (bottom right) depth maps.

Two snapshots taken during model simulation are shown in Figure 13, for t=0.50 at the beginning of the simulation and Figure 14, for t=1.75 at the end. The top portions of the figures represent accommodation modules while the bottom portions represent disparity modules. As time progresses, target ghost start to disappear and only real activity common to both accommodation and disparity stay active.

For this particular simulation, the distribution architecture for the depth perception model consisted of three Unix workstations, one used as console displaying graphics and providing user interactivity while the other two processed the actual modules, as shown in Figure 15.

Current benchmarks have shown that the distributed simulation of the Depth Perception model may run between 5 to 10 times as fast as the sequential one depending on overall machine and network load.

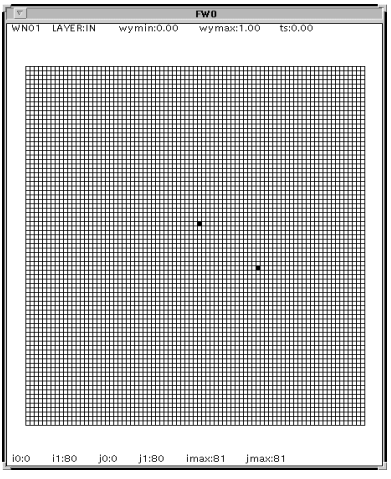


Fig. 11. The *Retina* module contains two output ports, *dp* and *ap*, for disparity and accommodation, respectively. The *Depth* module consists of an input port *sp*, receiving data from the *Retina*, a second input port *tp*, receiving input from the other *Depth* module, and an output port *mp*.

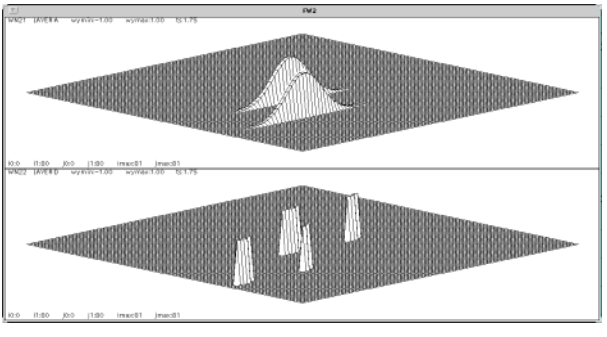


Fig. 12. The *Retina* module contains two output ports, *dp* and *ap*, for disparity and accommodation, respectively. The *Depth* module consists of an input port *sp*, receiving data from the *Retina*, a second input port *tp*, receiving input from the other *Depth* module, and an output port *mp*.

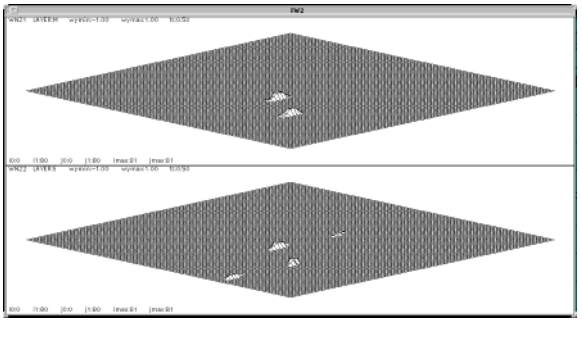


Fig. 13. Disparity *s* and accommodation *m* modules during simulation time 0.50.

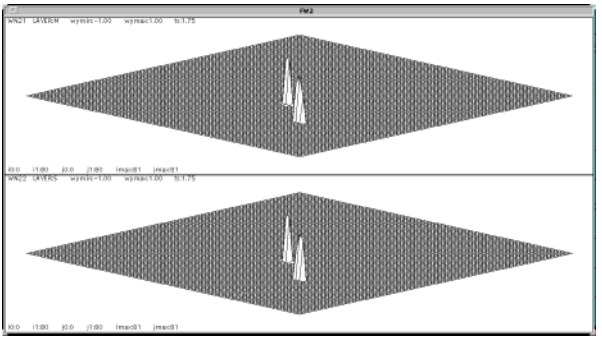


Fig. 14. Disparity *s* and accommodation *m* modules during simulation time 1.75.

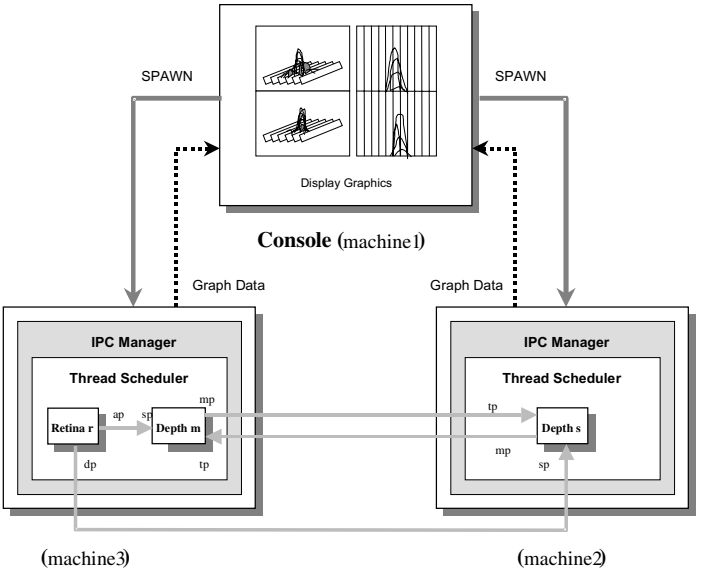


Fig. 15. Distributed simulation of the depth perception model under NSL/ASL.

5 Conclusions

This paper has shown the basic concepts involved in modular neural network development and distributed simulation in NSL/ASL. While a large number of neural models have been developed throughout the years in NSL, we are in the process of testing them under the NSL/ASL distributed architecture. Distributed simulation performance, to be made more precise as further benchmarks are carried out, should reflect good processing improvement not only in smaller models but especially in larger models as well. Additionally, we are in the process of putting together independently developed modules as part of more complex models involving a large number of modules. Previously, this was prohibitively expensive to either model or

simulate, particularly with workstations. For example, the depth perception model becomes just one component in the Frog's *Learning to Detour* model [17].

Other simulation systems have taken "parallel" approaches to deal with complex simulation, such as PGENESIS, NEURON[1], as well as previous versions of NSL. These versions tended not to be too "friendly" in their usage requiring special effort in the part of the modeler to adapt to the programming environment. On the other hand, the distributed NSL/ASL architecture provides a more accessible alternative to standalone parallel computation. Furthermore, in more sophisticated distributed environments a supercomputer can play a key role intensive module processing with graphics and command interaction taking place in a workstation.

Future extensions to the NSL/ASL distributed system involve how to troubleshoot it as well as optimize it. A reflective meta-level architecture [18] is being added to ASL providing monitoring capabilities [19] at the module and port communication level to improve overall simulation performance, *e.g. scheduling*, *deadlocks*, *load balancing*, *communication*, etc. Other extensions to NSL/ASL include linkage to the NSL Java version in a single heterogeneous Java/C++ distributed environment where we are considering other alternatives to PVM (as well as Pthreads) for inter-process communication particularly CORBA [20]. And while distributed computation is quite useful in improving processing performance, applications such as the control of remote robots [21] or the Web itself, have distribution as a primary requirement [22][2].

Ackowledgments. We thank the NSF-CONACyT collaboration grant (#IRI-9505864 in the US and #546500-5-C018-A in Mexico), the CONACyT REDII grant in Mexico, as well as the "Asociación Mexicana de Cultura, A.C.". Additionally we want to thank all students previously involved in the project in particular Salvador Mármol and Claudia Calderas.

References

1. Rall, W., Branching dendritic trees and motoneuron membrane resistivity, Exp. Neurol., 2:503-532, 1959.
2. Arbib, M.A., *The Metaphorical Brain 2: Neural Networks and Beyond*, pp. 124-126. Wiley Interscience, 1989.
3. McCulloch, W.S. & Pitts, W.H., A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bull. Math. Biophys*. 5:115-133, 1943.
4. De Schutter, E., A Consumer Guide to Neuronal Modeling Software. *Trends in Neuroscience*. 15:462-464, 1992.
5. Murre, J., Neurosimulators, in Michael Arbib (ed.) *The Handbook of Brain Theory and Neural Networks*, pp. 634-639, MIT Press, 1995.

[1] These two implementations are accessible from the Pittsburgh Supercomputer Center.

[2] All information regarding NSL and ASL as well as C++ downloads are available from "http://cannes.rhon.itam.mx" while the NSL Java version is obtainable from "http://www-hbp.usc.edu".

6. Bower, J.M. & Beeman, D., *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System*, TELOS/Springer-Verlag, 1998.
7. Hines, M.L. & Carnevale N.T., The NEURON simulation environment. *Neural Computation*, 9:1179-1209, 1997.
8. Leigthon, R.R. & Wieland, A.P. , The Aspirin/Migraines Software Package, in J. Skrzypek (ed.) *Neural Network Simulation Environments*, pp. 209-227, Kluwer, 1994.
9. Weitzenfeld, A., Alexander, A. & Arbib, M.A., *The Neural Simulation Language NSL, Theory and Applications*, MIT Press, 2000.
10. House, D., Depth Perception in Frogs and Toads: A study in Neural Computing, *Lecture Notes in Biomathematics 80*, Springer-Verlag, 1985.
11. Weitzenfeld, A. & Arbib, M., A Concurrent Object-Oriented Framework for the Simulation of Neural Networks, in Proc. of ECOOP/OOPSLA '90 Workshop on Object-Based Concurrent Programming, *OOPS Messenger*, 2(2):120-124, April, 1991.
12. Weitzenfeld, A., ASL: Hierarchy, Composition, Heterogeneity, and Multi-Granularity in Concurrent Object-Oriented Programming, *Proc. Workshop on Neural Architectures and Distributed AI: From Schema Assemblages to Neural Networks*, Oct 19-20, USC, October 19-20, 1993.
13. Arbib, M.A., Schema Theory. In Stuart Shapiro (ed.), *The Encyclopedia of Artificial Intelligence*, 2nd. Edition, 2:1427-1443, Wiley, 1992.
14. Yonezawa, A. & Tokoro, M., *Object-oriented concurrent programming*, MIT Press, 1987.
15. Geist, A., Beguelin, A., Dongarra J., Jiang, W., Mancheck, R. & Sunderam, V., *PVM: Parallel Virtual Machine, A User's Guide and Tutorial for Networked Parallel Computing*, MIT Press, 1994.
16. Lewis B. & Berg D.J., *Multithreaded Programming with Pthreads*, Sun Microsystems Press - Prentice Hall, 1998.
17. Corbacho, F. & Arbib, M.A., Learning Internal Models to Detour, *Society for Neuroscience*. Abs. 624.7, 1997.
18. Kiczales, G. & Paepcke, A., *Open Implementations and Metaobject Protocols*, Palo Alto Research Center, Xerox Corporation, 1996.
19. Gutiérrez, S. *Design of a Reflective Meta-level Architecture for NSL/ASL*, TR-99-01, Dept. Comp Eng, ITAM, Sept., 1999.
20. Mowbray, T. & Ruh, W., *Inside CORBA: Distributed Object Standards and Applications*, Addison-Wesley, 1998.
21. Weitzenfeld, A., Arkin, R.C., Cervantes-Perez, F., Olivares, R., & Corbacho, F., A Neural Schema Architecture for Autonomous Robots, *Proc. 1998 Int. Symposium on Robotics and Automation*, Dec. 12-14, Saltillo, Coahuila, Mexico, 1998.
22. Alexander, A., Arbib, M.A. & Weitzenfeld, A., Web Simulation of Brain Model, in A. Bruzzone, A. Uhrmacher and E. Page (eds.) Proc. 1999 Int. Conf. On Web-Based Modeling and Simulation , pp. 124-126. 31(3):29-33, Soc. Comp Sim, 1999.

Dynamical Behavior of an Electronic Neuron of Commutation

A. Padrón, J. L. Pérez, A. Herrera, and R. Prieto

Laboratorio de Ingeniería Neuronal, Centro de Instrumentos - UNAM
Apartado Postal 70 - 186, Coyoacán, C. P. 04510, México D.F.
Tel. : (01) 5622 8632, Fax: (01) 5622 8617.
E-mail: apadron@aleph.cinstrum.unam.mx

Abstract. In this work we present the design of an electronic model of a single commutation neuron and illustrate some of its dynamic behavior. This kind of electronic neuron shows changes in the activity of its potential when it is equal to threshold level constant. In particular, the neuron model developed presents commutation processes in its dynamical behavior. That is, the model is integrative as a leaky integrator below the threshold level and shoots when it reaches it; then the neuron's potential decays similar to an integrate and fire model. The electronic neuron can commute between, at least, two different kinds of oscillatory behavior. As a consequence of this, the neural response can be formed with different combinations of spikes and pulses.

1. Introduction

Neurons, whether real or artificially modeled, are basic processing elements of computational neural networks. Formalizing single – neuron computation has been inspired by two types of endeavors. For computer scientists and engineers, understanding the computation process performed by neurons is essential to the design of basic processing elements that will equip new models of brain - like computers. In contrast, computational neuroscientists have been exploring the biophysical mechanisms that enable neurons to "compute", in order to support their computational explanation of brain functions. In this way, computational ability and biophysical plausibility, respectively, are the two focuses of interest for researchers in these two different areas. However, it is clear that new insights into either of these aspects would inevitably benefit both.

Now, the current generation of artificial neural networks found in both areas makes use of highly simplified models of neurons, such as summation–and–firing models. The reason for the use of these simplified models is twofold. On one hand, many researchers believe that the complicated dynamic behavior and advanced functions of a neural network system are primarily the result of collective behaviors of all participating neurons, and are less relevant to the operational detail of each processing element. Assuming this, simple neuron models, such as summation–and–firing or

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 338—349, 2000.

leaky integrator models, are preferable to complicated neuron models, such as the compartment ones, because of their ease of implementation and understanding. On the other hand, there has been a lack of available models of single neurons that are both computationally powerful and biologically plausible as replacements of the current models. In fact, as a main source of novel neuron models, computational neuroscience is just tackling the principles of single – neuron signal and information processing.

In addition, the numerical simulation of neural network models, whether simplistic or realistically complex, is a key problem in the field because of their computational complexity. For this reason, a large number of software- and hardware-based neural network implementations have been developed in recent years that attempt to streamline the number of computations and take advantage of the inherent parallelism found within neural network architectures. Nowadays, we can distinguish three board areas of neural network implementation: 1) computer-based software, 2) electronics hardware, and 3) optical and opto-electronic implementations. In particular, electronic implementations have been primarily oriented to the design and construction of bus-oriented processors, coprocessors and VLSI circuit designs. Now, due to the fact that in this moment there are a lot of feasible techniques to create very complex digital circuits, analog implementations of neural networks are rare to find in the technical literature. At first sight, this seems strange since analog implementations have important advantages over digital ones. For example, complex and nonlinear operations can be performed in an analog circuit with much less devices than those needed in a digital circuit. However, the design, analysis, and verification of analog circuits is a very difficult task due, in part, to the inherent complexities of the basic techniques. In spite of this, it is now widely recognized in the neural network community that one of the most promising strategies for implementing neural network systems is through the use of electronic analog VLSI circuits.

In this work we present a slight modification to an electronic implementation of an integrative - and – firing neuron model. We show that the new neuron model has a richer dynamic behavior than that of the original model. We call this model the commutative neuron. This name comes from the fact that our electronic neuron can commute between different dynamic behaviors. In particular, we will show that the commutative neuron has a richer oscillatory behavior that the one exhibits by a typical integrative – and – firing neuron model.

2. Basic Neuron Model

In simple terms, an artificial neuron can be depicted as a signal processing element that receives n input signals $\boldsymbol{X}(t)$ in vector form and yields a scalar output signal $y(t)$. The input vector $\boldsymbol{X}(t)$ represents signals being transmitted from n neighboring neurons including self-feedback signals and/or outputs from sensory neurons. From a mathematical point of view, the signal-processing capabilities of an artificial neuron can be represented as a nonlinear mapping operation, F_{ne}, from the input vector $\boldsymbol{X}(t)$ to the scalar output $y(t)$; that is, F_{ne}: $\boldsymbol{X}(t) \rightarrow y(t)$. In addition, the neuronal nonlinear mapping function F_{ne} can be divided into two parts so called a) *confluence* and b) *nonlinear activation* operations. The confluence operation generates the dendritic and

somatic signals from the neural inputs. The nonlinear activation operation performs a nonlinear mapping on the somatic signal of the neuron to produce the axonic and terminal signals of the neuron. Normally, this activation operation is implemented in terms of sigmoidal functions, e. g., logistic functions.

The most common implementation of the basic artificial model is based on three elementary neuronal operations, two linear and one non-linear. The two linear operations model the dendritic and somatic operations found in the biological neurons, whereas the non-linear operation or activity function models axonic and terminal operations of the neuron. The first linear operation is implemented as an internal product between a vector of time dependent external signal and a vector of internal time independent neuronal parameters, the synaptic weights; the second linear neuronal operation is implemented as a leaky integrator type function characterized with a single time constant. On the other hand, the non-linear operation we had considered is a comparator function.

Finally, the neural operations are said to be polarized. That is, a neuron receives signals via dendrites, generating the dendritic signal, process and accumulate it in the soma, giving rise to the somatic signal, and producing and sending the response signal, formed from the axonic and terminal signals, to other neurons.

In this work we are interested to study a slight modification of this basic model in the following sense. We consider that the somatic and terminal signals interact in a reciprocally fashion.

Leaving of the neuron general model representation, Gupta (1994), which one is following presented figure (1), the structure of the neuron commutation model will be described. The general neuron model, it has inputs confluence operation and an activation operation to obtain an output. This neuron model can generally be described as a system of multiple inputs $\{x_1(t), x_2(t), x_3(t),..., x_n(t)\}$ that maps a space of R^n to a simple output (y(t)) in the space R^1, through a non-linear activation function Ψ. In the figure (1) the structure of the general neuron model is presented.

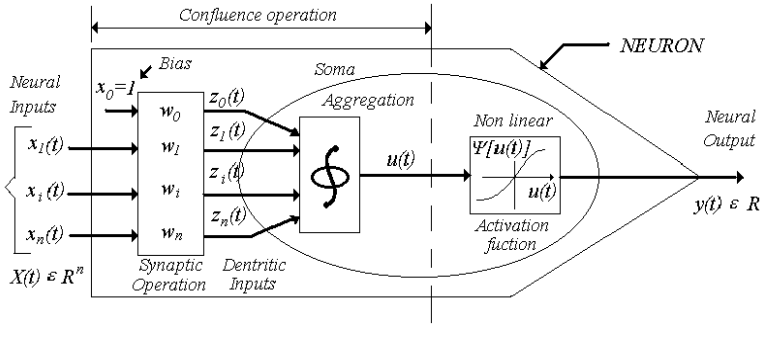


Fig. 1. *Structure of a general artificial neuron model.*

In figure (2), it presents the commutation element of integration stage. This element allows stopping instantaneously the integration activity and carrying out the commutation among oscillatory, commutation to no-oscillatory activity of artificial neuron model. The neuron dynamical behavior change producing spikes by means of the activation function also called threshold function, usually the hard–limiter function with a level threshold constant.

The modification of the neuron model is shown in figure (2). The commutation element consists on interrupt the integration activity, converting the behavior in only a pondered sum of the signals input, followed by the activation function. In this case an analogic switch as commutation element gives this interruption, thus it is producing the changes. The time constant of model has important role in integration activity and as consequence dynamical behavior, this constant will be a parameter τ given by integration configuration of commutation model

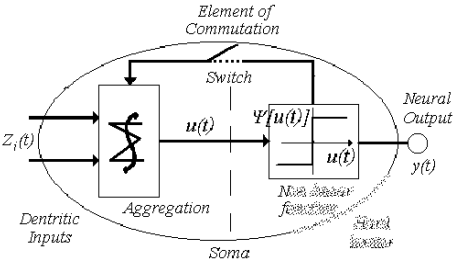


Fig. 2. *Element of commutation that instantaneously interrupt the stage of integration.*

3. Electronic Circuit of Commutation Neuron Model

The neuron circuit design is presented in figure (3), this show several stages of artificial commutation neuron model. It has a stage that receives external signal input, and it ponders by resistor (weight). Later it has an adder stage of pondered input, it is given by operational amplifier in adder configuration. It continues an integration stage that it is given by operational amplifier in integration configuration, the time constant of integration stage τ is approximately equal to 1 millisecond. Until here is confluence united with the aggregation operation of external inputs in the circuit.

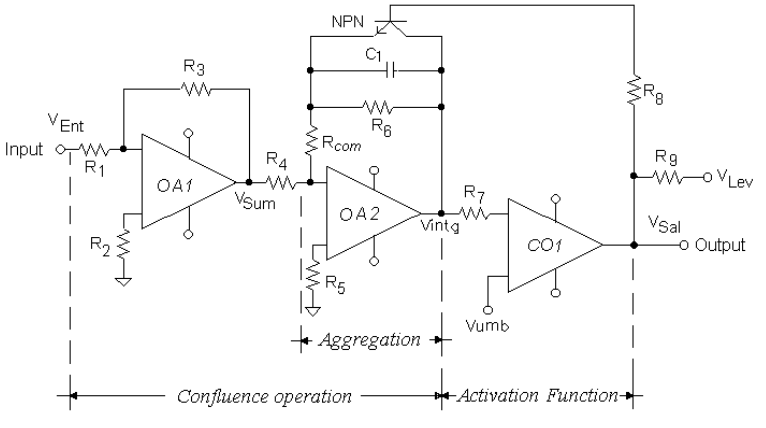


Fig. 3. *Schematic circuit of single commutation neuron model.*

The model has an activation stage that provides form to the neuron response with an operational amplifier in comparator configuration, (neuron activation function hard–limiter). The reference voltage (1V) of comparator is the voltage threshold level constant. In the same circuit has a commutation element, that is, analogic switch, between out and integration stage. We used a NPN transistor as switch in order to interrupt integration. This way we consider that the somatic and terminal signals interact together in a reciprocally fashion as a kind of feedback in the circuit. This circuit is tested with square periodic signal input and amplitude of 5V and 1 kHz of frequency, all resistors in circuit has the same values of 10[kΩ] the resistor R_{com} is parameter of commutation for dynamical behavior.

The input stage consists of a first operational amplifier (OA1) in adder configuration, which catches all input signals (external or internal, stimulus), and they are pondered by means resistors at input of this operational amplifier. This operational amplifier OA1 adder has a unitary amplification factor given by A=1, for that the response of this stage achieve an output voltage V_{Sum} given by pondered sum of input signals.

The second stage of this electronic circuit is conformed by operational amplifier (OA2) in integration configuration, the work of this stage is carrying out the integration of V_{Sum}, that is known as aggregation operation. The amplification factor for this stage is provided by the quotient among the resistances R_6 and R_4 o circuit, that which amplifies or it attenuates the response signal given by the output voltage V_{intg} after the commutation, it will be seen in the description of the following section. According to figure (1) together stages carry out the confluence operation.

The comparator CO1 is who carry out the activation operation of type step, according to this output voltage V_{Sal} it is obtained if the voltage V_{intg} is equal to the voltage V_{umb} of CO1. To obtain a level voltage to output of CO1 (V_{sal}), it is applied a voltage V_{Lev}.

These are typical elements of an artificial neuron electronically implanted in analogical circuits, but it lacks the element that it carries out commutation process. To obtain a commutation response in the circuit, a commutation element is used, for that task it is used a NPN transistor, which is in short state under level voltage of 700 [mV] approximately and in saturation state up 700 [mV] approximately. The NPN transistor is activated or it is saturated when in CO1 the Vintg is equal to voltage Vumb =1 [Volt] and the CO1 respond. The CO1 output voltage goes to transistor base, this voltage makes that the transistor is saturated and therefore it put in short at capacitor C_1 with resistance R_4 achieving this way almost instantly discharge of C_1.

When above is achieved, OA2 it becomes a signal follower for a moment with an amplification factor that causes a change in amplitude output voltage. The voltage Vintg is also shown affected in following way: when V_{intg} is equal to V_{umb} the V_{intg} it decays almost instantly to zero level or potential of rest, this activity is similar to integrate-and-fire behavior, thus at output it is observed a spike. Once capacitor C_1 is discharged, of the integration stage, the process repeats obtaining a train of spikes at output.

One of the later sections it will be shown how is the type of activity of this circuit when it is carried out a variation in resistor R_{com} of circuit and responses are presented. The resistor R_{com} is located in integration stage of circuit. Changing this parameter is affected amplification factor of AO2, for that several behaviors are presented in the circuit, it will see each other later on.

4. Description of the Electronic Circuit of Commutation Neuron Model

In this section a circuit description of the commutation neuron is presented by stages, according to its structure of the section 2, figure (2), and circuit schematic of the section 3, figure (3). The stages described are the following ones: adder, integration, comparator, and commutation element.

In the figure (4) an Amplifier Operational is observed in inverter adder configuration, to continue with the previous notation it is designated to input voltage by V_{Ent} and to output voltage of this stage by V_{Sum}. If it is considered several input signals at circuit and we denote them by V_a, V_b, and V_c, which are denominated external or internal stimuli in a neuron, and they enter to circuit through resistors R_a, R_b, and R_c respectively. Then description of adder OA1, is made by means of Kirchoff currents laws that circulate through its.

$$I_{Sum} + I_B = I_a + I_b + I_c \tag{1}$$

where: $I_{Sum} = \frac{V_{Sum}}{R_F}$, $I_a = \frac{V_a}{R_a}$, $I_b = \frac{V_b}{R_b}$, and $I_c = \frac{V_c}{R_c}$. With $I_B = 0$.

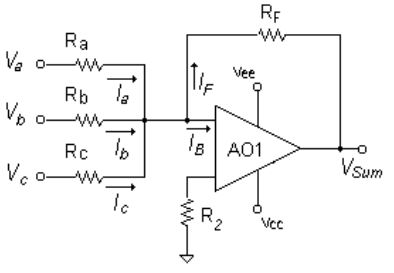


Fig. 4. *Operational amplifier in adder configuration.*

An arrangement of resistors in parallel is used at input of operational amplifier OA1, with follower's configuration, which carries out task of inputs sum, then according to the equation (1) one has:

$$V_{Sum} = -\left(\frac{R_F}{R_a}V_a + \frac{R_F}{R_b}V_b + \frac{R_F}{R_c}V_c\right) = -R_F\left(\frac{V_a}{R_a} + \frac{V_b}{R_b} + \frac{V_c}{R_c}\right) \tag{2}$$

If it is takes to $R_a = R_b = R_c = R$, then:

$$V_{Sum} = -\frac{R_F}{R}\left(V_a + V_b + V_c\right)$$

after: $A = \frac{R_F}{R} = 1$, that it is the gain of the circuit or amplification factor.

The output voltage V_{Sum} is equal to negative sum of all input voltages. For that,

$$V_{Sum} = -(V_a + V_b + V_c) \tag{3}$$

The second stage of circuit, figure (3), it consists of an OA2 in integration configuration, which is represented in figure (5). Description of this circuit is carried out in the same way that previous one.

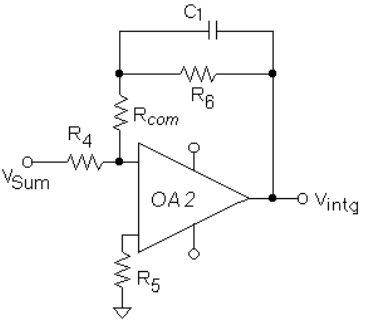


Fig. 5. . *Operational amplifier in integrative configuration.*

This way by means of the currents Kirchoff laws in the node of entrance V_4:

$$I_4 = I_B + I_1, \tag{4}$$

where I_B is relatively small, and this way $I_4 \approx I_F$. Remembering that relationship between current and voltage through a capacitor is,

$$i_c = C\frac{dV_C}{dt}, \tag{5}$$

Therefore:

$$\frac{V_{Sum} - V_4}{R_4} = C_1 \frac{d}{dt}(V_4 - V_{\text{int}\,g}); \tag{6}$$

however $V_4 \approx 0$, for that gain of amplifier is high, of here that,

$$\frac{V_{Sum}}{R_4} = C_1 \frac{d}{dt}(-V_{\text{int}\,1}),$$

then separating variables and integrating, one has:

$$\int \frac{V_{Sum}}{R_4} dt = \int C_1 \frac{d}{dt}(-V_{\text{int}\,g})dt = C_1(-V_{\text{int}\,g}) + V_{\text{int}\,g}\Big|_{t=0},$$

with this:

$$V_{\text{int}\,g} = -\frac{1}{R_4 C_1}\int V_{Sum} dt + C, \tag{7}$$

with C equal to an integration constant. Which it can also express with following expression for voltage V_{intg},

$$V_{\text{int}g} = -\frac{1}{R_4 C_1}\int_0^t V_{Sum}\,dt\,, \tag{8}$$

if $f_a = \dfrac{1}{2\pi R_6 C_1}$, is frequency in terms of $R_6 C_1$ and,

$f_b = \dfrac{1}{2\pi R_4 C_1}$, is frequency in terms of $R_4 C_1$;

which should be selected in such way that $f_a < f_b$. For example if $f_a = f_b / 10$ then $R_6 = 10R_4$. The signal will be integrated appropriately if $R_6 C_1 \leq \tau$, where, τ it is the period (constant of integration) and expressing it otherwise,

$$R_6 C_1 = \frac{1}{2\pi f_a}\,. \tag{9}$$

On the other hand in the same OA2 it has that for Kirchoff,

$$I_4 = I_B + I_c \tag{10}$$

where I_B is small relatively, and this way $I_4 \approx I_c$. Therefor:

$$\frac{V_{Sum} - V_4}{R_4} = \frac{V_4 - V_{In\,\text{tg}}}{R_c + R_7}\,. \tag{11}$$

To find voltage V_{intg}, it is has:

$$V_{\text{int}g} = -\frac{R_c + R_7}{R_4} V_{Sum} \tag{12}$$

Then with the equations (7) and (12), joining both analyses voltage V_{intg} expression is obtained at output of OA2,

$$V_{\text{int}g} = -\frac{R_c + R_7}{R_4} V_{Sum} - \frac{1}{R_4 C_1}\int V_{Sum}\,dt + C \tag{13}$$

which will be conditioned by response of comparator CO1 of the figure (3), achieving that some terms of the equation (13) for an instant they become zero.

The activation stage is conformed by comparator CO1. A comparator, as its name indicates it, it compares a voltage signal on one input pin of amplifier with a voltage signal well-known, called reference on other input pin of operational amplifier. This simple way, it is not more than an amplifier "open-loop", with two input analogical and output binary; output could be voltage (+) or voltage (-) of saturation, depending on that input is higher and of configuration for level voltage.

In the figure (6) it is shown integrated circuit CO1 used as comparator of circuit. A fixed threshold voltage V_{Umb} of +1 V is applied to input (-) of CO1, and at other input (+) it is applied time varying voltage, V_{intg} in this case. This is reason why the

configuration it is called non-inverter comparator. When $V_{intg.}$ is smaller than V_{umb} output voltage V_{CO1} is in 0 V, because the voltage to input (-) it is higher than input (+). Thus,

$$V_{CO1} = \begin{cases} V_{+Lev} & \text{if} \quad V_{\text{int}\,g} > V_{umb} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

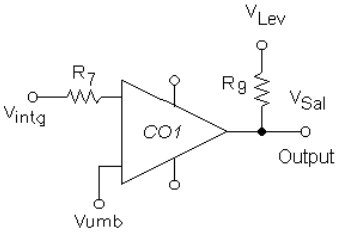


Fig. 6. *Stage of Activation CO1 comparator.*

When it happens at inverse, namely when V_{intg} is higher than V_{Umb}, (+) input it becomes positive with regard to (-) input, and V_{CO1} taking value of $+V_{Lev}$ ($\cong +V_{cc}$). This way V_{CO1} changes from saturation level to other when $V_{intg} \cong V_{Umb}$. For any time the voltage V_{CO1} form is higher or smaller than threshold voltage V_{Umb}. The comparator is sometimes called detecting of voltage level, because, for wanted value of V_{Umb}, it can detect input voltage level V_{intg}.

The voltage V_{CO1} is equal to output V_{Sal}, this voltage is the response of electronic circuit of commutation neuron.

Figure (7) shows configuration of circuit between OA2 and CO1 for short-saturation operation of NPN transistor in the circuit, that it achieves commutation behavior in the electronic circuit.

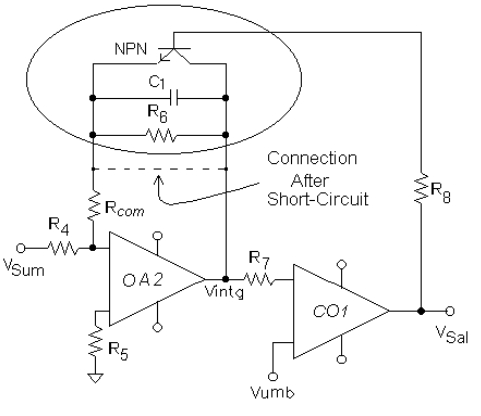


Fig. 7. *Relationship between OA2 and CO1 for commutation process.*

Output voltage of CO1 denoted by the voltage V_{CO1}, also achieves activation or saturation at NPN transistor of figure (7), this signal causes changes in voltage V_{intg} given by the equation (13).

$$V_{\text{int } g} = -\frac{R_c + R_7}{R_4} V_{Sum} - \frac{1}{R_4 C_1} \int V_{Sum} dt + C$$

What happens is that when the transistor working (in short-saturation), it set at capacitor C_1 and R7 resistor in short, therefore in previous equation the integrating term and R7 resistor are zero, then:

$$V_{\text{int } g} = -\frac{R_c}{R_4} V_{Sum} \ , \tag{15}$$

that it can rewrite like, $V_{\text{int } g} = -AV_{Sum}$, where A is instantaneous amplification factor provided by quotient among R_c and R_4 in OA2.

5. Dynamical Behavior of Commutation Neuron Model

In order to study this electronic circuit we have a dynamical behavior when the periodic input is presented, input is integrated until the potential integrative reaches reference voltage of comparator and response with 5V of amplitude. In this instant the NPN transistor is polarized and yield short-circuit in integrator stage, and the potential integrative decrease instantaneously cause capacitor discharge, which this stage is converted in a follower configuration by R_{com} resistor in some cases instantaneously. Figure (8,9,10) show responses of our single commutation neuron model, changing the R_{com} values. In each graph we show the input, integrative potential and output. The R_{com} values are: R_{com}=1[kΩ], R_{com}=100[Ω] and R_{com}=1[Ω] for figures (8,9,10) graphs respectively.

First case show an oscillatory dynamical behavior in its response, spikes instantaneous, due at big value of R_{com}. This cause that the capacitor discharged and charged again rapidly around threshold level, without that integrative potential decrease to initial conditions and repeat the process.

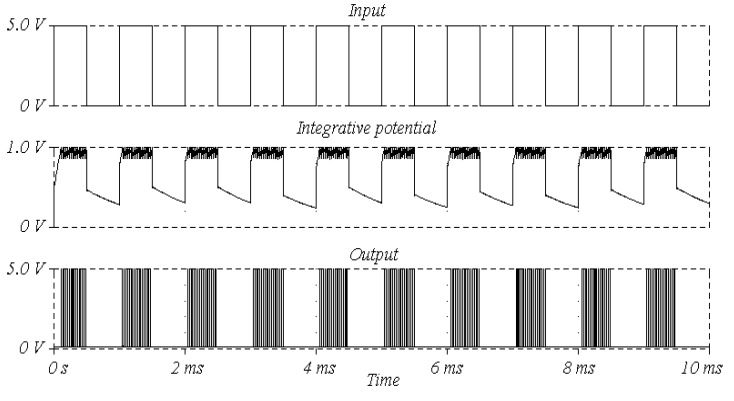


Fig. 8. *Oscillatory Response, R_{com} = 1 [kΩ].*

For second case, we have a interval value of R_{com} in which yield a commutation response between spikes and pulses for same periodic input. That is, in a small time the capacitor is discharged and charged again but the integrative potential is saturated in threshold level after this time, yielding a pulse output.

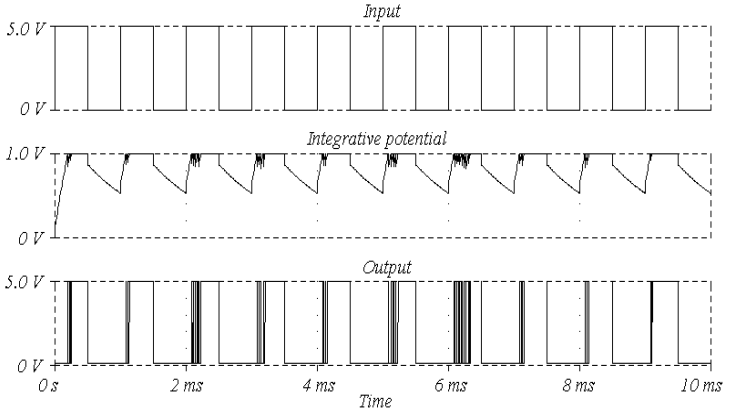


Fig. 9. *Commutation Neuron Response, R_{com} = 100 [□□.*

At least case when we have a small value of R_{com} is obtained a no-oscillatory behavior. This cause that capacitor stay charged and integrative potential stay saturated in threshold level, yield only a pulse output.

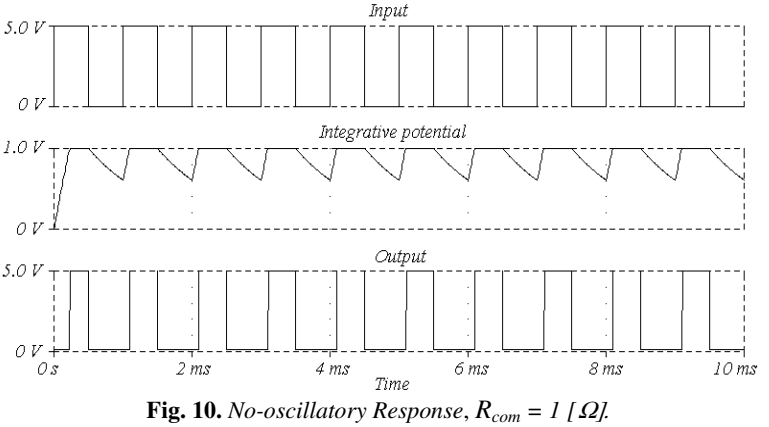


Fig. 10. *No-oscillatory Response, R_{com} = 1 [Ω].*

6. Conclusions

Using our model, we obtain the transition between Oscillatory to Commutation and No-Oscillatory responses, employing the principles Leaky Integrate and integrate-and-fire models, through of simple change of parameter R_{com} in the electronic circuit of figure (2). The results obtained for the single commutation neuron in its dynamic behavior shown different potential activities for the same input periodic signal. Is

important to say that the time constant of integration stage in electronic circuit stay constant. The advantage of this electronic model, as a basic processor, is that we can produce a transition dynamical behavior between oscillatory and no-oscillatory behavior of artificial single neurons with only a change in same circuit.

Dynamic systems were examined, simulating numerically integrate a fire neuron model and their derivations, looking for alternatives to find a mathematical expression of this model. Does not exist in the literature generic mathematical models of integrate and fire model, much less for commutation model. In this work we wanted to present different approach at biological model given by dynamic systems in form of differential equations. This circuit with commutation processes simulates several models of artificial isolated neurons, without having to use different mathematical expressions, the commutation is carried out with only a parameter change.

References

[1] M. Stremmler, "A Single Spike Suffices: The Simplest Form of Stochastic Resonance in Model Neurons", Network Computation in Neural Systems, vol. 7, pp. 687-716, November 1996.

[2] M. Bove, et al., "Dynamics of Networks of Biological Neurons: Simulation and Experimental Tools", Algorithms and Architectures, USA, 1998.

[3] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve", J. Physiol., vol. 117, pp. 500-544, 1952.

[4] W.S. McCulloch and W. A. Pitts, "A logical calculus of the ideas imminent in nervous activity". Bulletin of Mathematical Biophysics, 5:115-133, 1943.

[5] A. Cichocki, R. Unbehauen, "Neural Networks for Optimization and Signal Processing". Ed.Wiley, EUA, 1993.

[6] D. Tal, E. L. Schwartz, "Computing with the Leaky Integrate - and - Fire Neuron: Logarithmic

Computation and Multiplication", Neural Computation, Vol. 9, Num. 2, 305-318, U.S.A., 1997.

[7] A. Padrón, "Diseño de circuitos electrónicos para generar funciones de activación empleadas en redes neuronales artificiales", Tesis de Maestría en Ingeniería Eléctrica opción Electrónica, DEPFI-UNAM, Marzo, 1998.

[8] C. Mead, "Analog VLSI and Neural System", Ed. Addison Wesley, USA, August, 1989.

[9] M.M. Gupta, D. H. Rao, "Neuro-Control Systems: A Tutorial", Ed. IEEE PRESS, New York, 1994.

[10] A. Padrón, J. L. Pérez, A. Herrera, "Procesos de Conmutación en Una Neurona Artificial Aislada" Memorias SOMI XIII, Congreso de Instrumentación, Ensenada, B.C.N., pp. 275-279, Octubre 1998.

Training Neural Networks Using Non-standard Norms – Preliminary Results

Angel Kuri Morales

Centro de Investigación en Computación
Instituto Politécnico Nacional
Unidad Profesional Adolfo López Mateos
Zacatenco
México 07738, D.F.
akuri@pollux.cic.ipn.mx
5729-6000 ext 56547
fax: 5729-6000 ext 56607

Abstract. We discuss alternative norms to train Neural Networks (NNs). We focus on the so called Multilayer Perceptrons (MLPs). To achieve this we rely on a Genetic Algorithm called an Eclectic GA (EGA). By using the EGA we avoid the drawbacks of the standard training algorithm in this sort of NNs: the backpropagation algorithm. We define four measures of distance: a) The mean exponential error (MEE), b) The mean absolute error (MAE), c) The maximum square error (MSE) and d) The maximum (supremum) absolute error (SAE). We analyze the behavior of an MLP NN on two kinds of problems: Classification and Forecasting. We discuss the results of applying an EGA to train the NNs and show that alternative norms yield better results than the traditional RMS norm.

1 Introduction

Neural networks (NNs) have found innumerable applications in science, economics, engineering, etc. One very popular alternative has been the one arising from the feedforward interconnection of simple perceptrons (the Multi Layer Perceptron network or MLP). This sort of NN is usually trained applying the so-called backpropagation algorithm: a generalization of the delta rule [RUME86] which allows the weights of the hidden layers to be found. The backpropagation algorithm (BA) has proven to be successful in many applications and continues to be one of the favorite training methods for the MLP. In fact, it is considered to be "the" training method for MLPs which, in this context, should more aptly be called the Backpropagation MLPs (BMLPs).

Despite its relative success, the backpropagation algorithm imposes certain restrictions in the process of training. For example: a) The activation function must be differentiable, b) The derivative should be, preferably, easily obtainable (because the main computational burden of the training process lies in its calculation), c) The global network error to minimize is $E = \frac{1}{2}\sum_i (z_i - o_i)^2$ (which, as will be discussed, is not necessarily the best measure of distance), d) There are several parameters which

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 350—364, 2000.

have to be determined *a priori* (for example, the learning rate, the momentum, the weight decay, the learning rate decay, the pruning strategy, the relevance threshold and the learning strategy, among others). To the restrictions due to the BA, we must add another one, e) The basic architecture of the NN is arbitrarily defined (number of layers, neurons per layer, activation function,etc.). Finally and most importantly, f) The backpropagation algorithm does not guarantee convergence (i.e. a BMLP may not be properly trained in some cases).

The problem of training the NN and finding the best architecture are, both, optimization problems. In the first case we want to minimize a measure of distance (i.e. the accuracy with which a NN is able to *generalize* from a set of training data) whereas in the second we want to maximize the efficiency of the NN. In the present paper we propose the application of a Genetic Algorithm (GA) to the problem of properly training an MLP without the need to resort to the aforementioned BA.

GAs have been used in the past to, somehow, train NNs and to determine their architectures. For instance, Montana and Davis [MONT89] have evolved the weights of the NN; Miller, Todd and Hegde [MILL89] have evolved the network's architecture using direct encoding and Kitano [KITA90] used grammatical encoding for the same purpose. Further treatment of these issues may be found in [YAOX93] and [ROOI96]. These works, however, have not addressed how appropriate the RMS norm is in such endeavours, nor have they explored radical variations to the traditional GA (the so-called *Simple* GA). Here we assume that the SGA is not the best alternative and rely on previous work to make our choice of a more efficient and sturdier GA.

GAs are tools known to be able to tackle virtually any optimization problem. It has been mathematically proven [RUDO97] that an *elitist* GA will find the best solution, although the time involved is not bound in general. Goldberg [GOLD87] has identified a source of GA-hardness in the so called *deceptive* problems (and schemas) while Mitchell et al [MITC92] have identified *spurious correlation* as another such source. If we want to overcome both limitations, we have to abandon the traditional *Simple Genetic Algorithm* (SGA) discussed, originally, in [HOLL75] and replace it with a "stronger" version. The work we have conducted relies on the so called *Eclectic Genetic Algorithm* (EGA). A full discussion of the EGA lies outside the scope of this work. However, the interested reader may find a detailed description in [KURI99]. Suffice it to say that the EGA incorporates the following: a) Full elitism, b) Deterministic selection, c) Annular crossover, d) Probabilistic activation of a random mutation hillclimber (RMHC), e) Self-adaptive determination of P_C (crossover probability), P_M (probability of mutation), N (number of offspring) and η_τ (probability of activation of the RMHC). The EGA has been proved to perform adequately in the optimization of a set of GA-hard set of functions and has been applied in solving various non-numerical optimization problems in which the SGA fails to find an acceptable solution. Using the EGA we have been able to train a MLP in a way that compares favorably with the BA and, furthermore, relieves the designer from having to be concerned about (a), (b), (c), (d) and (f) above.

In part 2 we describe the methodology we followed to train a MLP using the EGA. In part 3 we report our results by establishing a comparison between what we call a *Genetic Neural Network* (GNN) and a traditional BMLP and, finally, in part 4 we offer our conclusions.

2 Methodology

To achieve the training of a MLP we encoded the weights in a genome whose length depends on the topology of the network, the pre-determined weights' range and the desired precision. We tested the genetically trained NNs (GNNs) using three sets of data pertaining to the following problems: a) A logistic problem (i.e. classification of data in two sets); b) A clustering problem (i.e. classifying data in n (n=3) sets) and c) A forecasting problem (i.e. learning the behavior of a system from a set of data). In all cases data were further divided in two sets: a) A training set (Tr) and b) A test set (Ts). The GNN is trained using Tr and the best network is selected from the weight set with the best performance over set Ts. The basic characteristics of GNNs applied to the three data sets are illustrated in table 1.

The presentation layer had one neuron for each input plus a bias neuron. The hidden layer consisted of the indicated neurons (2, 5, 5) plus a bias neuron. Hence, the number of connections (weights) N in the network is given by:

$$N = \sum_{i=1}^{L-1} (n^l + 1)(n^{l+1})$$

where

$$L = number of layers$$
$$n^l = number of neurons in layer "l"$$

Every connection's value (w_i) was represented in weighted binary with 32 bits: a sign bit and 31 binary digits (the most significant bit stood for 2^2. Therefore, weights' range is

$$-(8 - 2^{-28}) \leq w_i \leq +(8 - 2^{-28})$$

Weights were, initially, randomly set to values between -0.625 and +0.625. It is known that BMLPs generalize better when initial weights are small. An example of final settings for the weights is shown in table 2. The left column displays an astersik ("*") where the weight corresponds to a bias neuron. W[a](b,c) denotes the weight from layer "a", outcoming from neuron "b" onto neuron "c" in layer "a+1". Table 2 illustrates the fact that weights may take final values spanning the whole allowed range. For instance, W[2](2,1) lies close to the highest possible value, whereas W[2](2,2) lies close to the smallest. On the other hand, the final value of W[1](3,1) is much smaller than its initial (random) one.

The user of the EGA does not need to define the values of P_c (the probability of crossover) nor P_m (the probability of mutation); the algorithm dynamically adjusts these throughout the run. In all cases the population size N_P was set to 15. This unusually small population size was selected because the EGA induces very fast convergence of the network which impairs its generalization power. By using a small N_P we were able to achieve much better generalization. Of course this had the additional advantage of reducing training time. The number of generations for which the EGA was allowed to run was 120, 150 and 500 for the logistic, forecasting and classification problems, respectively.

Table 1. Network Architectures for the Three Problems

ARCHITECTURE	PROBLEM		
	Logistic	**Clustering**	**Forecasting**
Layers	**3**	**3**	**3**
Neurons in Presentation layer	3	13	12
Neurons in Hidden layer	2	5	2
Transfer Function	Sigmoid	Sigmoid	Tanh
Neurons in Output Layer	2	3	1
Transfer Function	Sigmoid	Sigmoid	Linear
Connections	14	88	29
Genome Size	448	2816	928
Elements in the Training Set	54	160	108
Elements in the Test Set	12	18	12

Table 2. Final Weight Settings (Example)

Bias Neuron	Weight	Value
*	W[1](1, 1)	-1.8574
*	W[1](1, 2)	2.9398
	W[1](2, 1)	3.8708
	W[1](2, 2)	-4.5245
	W[1](3, 1)	-0.0062
	W[1](3, 2)	0.3795
	W[1](4, 1)	-0.2853
	W[1](4, 2)	-1.5044
*	W[2](1, 1)	-1.5631
*	W[2](1, 2)	2.5868
	W[2](2, 1)	7.9438
	W[2](2, 2)	-7.7568
	W[2](3, 1)	-7.7481
	W[2](3, 2)	3.6868

2.1 Selected Norms

As already mentioned, one of the main issues of this work is to determine alternative ways to define a "best" NN. Of course, in all cases, our final measure of adequateness is how well the trained NN is able to perform when confronted with new data. But the error to minimize when training the NN it is not generally considered a parameter. In BMLPs backpropagation is, of course, assumed. Here we take advantage of the versatility of the evolutionary techniques to explore other alternatives.

In order to test the GNN we considered four different norms. In the GNNs it makes no difference whether the defined distance has "adequate" mathematical properties (for instance, distance MAE below is not continuously differentiable). In what follows, O(i) denotes the output from neuron i in the L-th layer; Z(i) denotes the desired output; *s* is the number of elements in the data set. The errors to minimize were defined as follows:

$$MEE = \frac{1}{s}\sum_{s}\sum_{i=1}^{n^L} 10^{|O(i)-Z(i)|}$$

$$MAE = \frac{1}{s}\sum_{s}\sum_{i=1}^{n^L} |O(i)-Z(i)|$$

$$MSE = \frac{1}{s}\sum_{s}\sum_{i=1}^{n^L} (O(i)-Z(i))^2$$

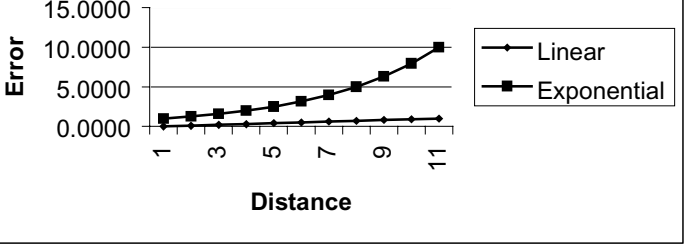
$$SAE = \max_{s,i} |O(i)-Z(i)|$$

In these definitions data are assumed to be normalized into the interval [0, 1]. The errors were considered over the whole data set (s) as opposed to optimizing the network for each sample. This strategy is usually called "cumulative optimization" in traditional BMLPs.

MEE (Mean Exponential Error).

This metric was defined with the purpose of giving the EGA more relevant information when finding better individuals. That is, if $\left\langle |O(i)-Z(i)| \right\rangle_{t+1} < \left\langle |O(i)-Z(i)| \right\rangle_t$ then $\left\langle 10^{|O(i)-Z(i)|} \right\rangle_{t+1} - \left\langle 10^{|O(i)-Z(i)|} \right\rangle_t$ is larger than $\left\langle |O(i)-Z(i)| \right\rangle_{t+1} - \left\langle |O(i)-Z(i)| \right\rangle_t$. This difference is illustrated in figure 1. If the EGA is able to identify the correct schemas more efficiently, then it will find the better weights faster and more adequately. The norm is defined in a way that it averages all the individual errors for the full set of samples. This way of considering the problem as a "solid" unit is easily achievable in a GA.

Figure 1. Illustration of the Exponential Norm



MSE (Mean Square Error).

This norm is analogous to the one minimized by the BA.

SAE (Maximum Absolute Error).

This is the counterpart of the typical minimax (L_∞) norm. We demand of the algorithm, however, that it finds the minimum maximum absolute error for all the expected outputs for all the s samples in the training set.

2.2 The Problems

The three problems we tackled are actual "real world" problems. The two classification ones stem from the characterization of three wine vintages. For each vintage the wine is characterized by 13 chemical components. The combination of these 13 components determines to which vintage a particular wine sample belongs. The forecasting problem refers to the behavior of the ticket sales of a commercial airline during a one-year period. In this case we used the sales in the last 12 months to determine the sales of the 13^{th} month. This information is represented in a 13 column table. The first 12 columns display ticket sales for months t-12, t-11,...,t-1. The last column displays the sales in month t. Therefore, we wish to forecast the values of the t^{th} month as a function of the 12 preceding ones, i.e. *sales (t) = sales(t-1,...,t-12).*

2.2.1 Logistic Problem

In the logistic problem we randomly selected a subset of size 3 out of the 13 features and two of the three vintages from the wine classification problem. A partial view of the training set is shown in table 3. In this set a "1" in the "Wine" column denotes

Table 3. A Sample of the Training Set for the Logistic Problem

Feature 1	Feature2	Feature 3	Wine A	Wine B
0.8421	0.1917	0.5722	1	0
0.5711	0.2055	0.4171	1	0
0.5605	0.3202	0.7005	1	0
0.8789	0.2391	0.6096	1	0
0.5816	0.3656	0.8075	1	0
0.3421	0.0711	0.4920	0	1
0.6947	0.1008	0.2995	0	1
0.3526	0.0771	0.4278	0	1
0.3000	0.1403	0.6257	0	1
0.3526	0.0929	0.6417	0	1

membership. Accordingly, the columns "Wine A" and "Wine B" may only take the values 0 or 1 and, furthermore, no two 1s nor two 0s may appear in any one row. The outputs from our NNs may take any value between 0 and 1. We, therefore, mapped the largest value to 1; the smallest to 0.

2.2.2 Clustering

In this problem we considered the full set of wine features mentioned in the preceding paragraphs. A partial view of the training set is shown in table 4. As above, membership is denoted with "1"; therefore, only one "1" may appear in any given row. Since the outputs from our NNs may take any value between 0 and 1 we mapped the largest value to 1; the smaller ones to 0.

2.2.3 Forecasting

The data sample from the training set for this problem displays the expected behavior: since values are displaced in time, the same value repeats itself diagonally. In table 5 we have included a sample of the training set where we have boldfaced one such value. One striking feature of NNs in general is that they are not susceptible to this kind of obvious linear dependence which, in other methods, seriously hampers the mathematical stability of the models.

Table 4. A Sample of the Training Set for the Clustering Problem

Feature 1	Feature 2	Feature 3	...	Feature 13	Wine A	Wine B	Wine C
0.8421	0.1917	0.5722	...	0.5613	1	0	0
0.5711	0.2055	0.4171	...	0.5506	1	0	0
0.5605	0.3202	0.7005	...	0.6469	1	0	0
0.8789	0.2391	0.6096	...	0.8573	1	0	0
0.3421	0.0711	0.4920	...	0.2867	0	1	0
0.6947	0.1008	0.2995	...	0.2511	0	1	0
0.3526	0.0771	0.4278	...	0.1013	0	1	0
0.3000	0.1403	0.6257	...	0.0549	0	1	0
0.4868	0.4447	0.5561	...	0.1797	0	0	1
0.4684	0.3103	0.5561	...	0.2011	0	0	1
0.4395	0.5553	0.5348	...	0.2297	0	0	1
0.4132	0.3399	0.4492	...	0.2974	0	0	1

Table 5. A Sample of the Training Set for the Forecasting Problem

t-12	t-11	t-10	t-9	t-8	t-7	t-6	t-5	t-4	t-3	t-2	t-1	Sales
0.32	0.35	0.40	0.38	0.34	0.42	0.48	0.48	0.43	0.35	0.28	0.36	**0.37**
0.35	0.40	0.38	0.34	0.42	0.48	0.48	0.43	0.35	0.28	0.36	**0.37**	0.38
0.40	0.38	0.34	0.42	0.48	0.48	0.43	0.35	0.28	0.36	**0.37**	0.38	0.46
0.38	0.34	0.42	0.48	0.48	0.43	0.35	0.28	0.36	**0.37**	0.38	0.46	0.41
0.34	0.42	0.48	0.48	0.43	0.35	0.28	0.36	**0.37**	0.38	0.46	0.41	0.43
0.42	0.48	0.48	0.43	0.35	0.28	0.36	**0.37**	0.38	0.46	0.41	0.43	0.43
0.48	0.48	0.43	0.35	0.28	0.36	**0.37**	0.38	0.46	0.41	0.43	0.43	0.49
0.48	0.43	0.35	0.28	0.36	**0.37**	0.38	0.46	0.41	0.43	0.43	0.49	0.49
0.43	0.35	0.28	0.36	**0.37**	0.38	0.46	0.41	0.43	0.43	0.49	0.49	0.43
0.35	0.28	0.36	**0.37**	0.38	0.46	0.41	0.43	0.43	0.49	0.49	0.43	0.35
0.28	0.36	**0.37**	0.38	0.46	0.41	0.43	0.43	0.49	0.49	0.43	0.35	0.29

3 Results

There are two measures of how successful a training method is. One is the training index (TI) which we define as the ratio between the number of correct values given by the trained NN when presented with the elements in the training set and the number of elements in such set. The other is the generalization index (GI) which we

define as the ratio between the number of correct values given by the trained NN when presented with the elements in the test set and the number of elements in such set. The TI measures how well a NN is able to learn. The GI measures how well a NN is able to establish a general model for a given problem. Thus, to determine the relative success of two training methods we found both TI and GI for the GNNs and a BMLP.

We conducted three sets of runs: one for each kind of problem. We trained the NN in 5 different ways in each case: one for each of the defined norms and one for the BMLP. The BMLP was trained using the following parameters: 1) Learning rate = 0.1, b) Momentum = 0.1, c) Weight Decay = 0.999999, d) Learning rate decay = 0.999999. The topologies were identical to those shown in table 1. Weights were randomly initialized in the range between –0.01 and +0.01. We consider one generation in the GNN to be equivalent to three epochs (the presentation of a full data set) in the BMLP. This consideration is justified by the fact that every generation in the GNN consisted of 15 individuals (where testing one individuals is equivalent to presenting a NN with the whole data set) but that in a GNN no backpropagation is necessary. Since the BA accounts for most of the computational cost of training the network we assume that one feedforward-backpropagation cycle is equivalent to 5 feedforward error calculations. Hence, 15 individuals in a GNN are roughly equivalent to 3 epochs in a BMLP. Accordingly, the BMLP was trained for 360, 450 and 1500 for the logistic, forecasting and classification problems, respectively.

3.1 Classification

Both the logistic and the clustering problems are classification problems. Hence, we obtained the behavior for: a) The clustering problem, b) The logistic problem and c) An average classification problem. In table 6 we display the performance of a) The NN trained genetically using the four defined norms and b) A BMLP trained as described above.

This preliminary results seem to be quite interesting. To begin with, it turns out that the GNNs trained with MSE norm generalizes worse than the BMLP, which was trained with an analogous norm. The explanation of this fact may lie in the EGA which, as already pointed out may converge too fast to allow generalization. One second conclusion is that the MAE and SAE norms yield the best Gis; better than the BMLP. The MEE norm, designed to allow the EGA to perform more efficiently, performed poorly. We may also observe that the logistic problem turned out to be more difficult to solve than the clustering one. This comes as no surprise since the data set in this case is artificial. We should remember that it was obtained from the partial consideration of elements and groups from the clustering problem. Since we trimmed the number of features from 13 to 3 it is a tribute to the NN model that it was able to adapt to the problem. For the purposes of this study, however, the relevant issue is that, again, the GNN trained with the SAE norm performs better than any other NN. When considering the TI, it is not surprising that all the methods performed rather well in the clustering case. Again the best norm was SAE (99.38%) which tied with MAE. But the other norms performed always above 90%. In the logistic case results were of lower quality. Here MEE was definitely poor (70.32%). But, again, the best performer was SAE which, even in this case, reached a TI of 94.13%.

Table 6. Performance of Different Training Methods for Classification Problems (Genetic and Backpropagation)

TEST SET (18/12 ELEMENTS IN THE SET)					
	Number of Correct Guesses in Clustering Problem	Generalizatio n Index	Number of Correct Guesses in Logistic Problem	Generalization Index	Average
Exponential	11	61.11%	7	58.33%	59.72%
Mean Absolute	14	77.78%	6	50.00%	63.89%
Mean Square	12	66.67%	8	66.67%	66.67%
MiniMax	14	77.78%	9	75.00%	76.39%
BMLP	13	72.22%	9	75.00%	73.61%
TRAINING SET (160/54 ELEMENTS IN THE SET)					
	Number of Correct Guesses in Clustering Problem	Training Index	Number of Correct Guesses in Logistic Problem	Training Index	Average
Exponential	148	92.50%	26	48.15%	70.32%
Mean Absolute	159	99.38%	43	79.63%	89.50%
Mean Square	146	91.25%	51	94.44%	92.85%
MiniMax	159	99.38%	48	88.89%	94.13%
BMLP	155	96.88%	44	81.48%	89.18%

This preliminary results seem to be quite interesting. To begin with, it turns out that the GNNs trained with MSE norm generalizes worse than the BMLP, which was trained with an analogous norm. The explanation of this fact may lie in the EGA which, as already pointed out may converge too fast to allow generalization. One second conclusion is that the MAE and SAE norms yield the best Gis; better than the BMLP. The MEE norm, designed to allow the EGA to perform more efficiently, performed poorly. We may also observe that the logistic problem turned out to be more difficult to solve than the clustering one. This comes as no surprise since the data set in this case is artificial. We should remember that it was obtained from the partial consideration of elements and groups from the clustering problem. Since we trimmed the number of features from 13 to 3 it is a tribute to the NN model that it was able to adapt to the problem. For the purposes of this study, however, the relevant issue is that, again, the GNN trained with the SAE norm performs better than any other NN. When considering the TI, it is not surprising that all the methods performed rather well in the clustering case. Again the best norm was SAE (99.38%) which tied with MAE. But the other norms performed always above 90%. In the logistic case results were of lower quality. Here MEE was definitely poor (70.32%). But, again, the best performer was SAE which, even in this case, reached a TI of 94.13%.

3.2 Forecasting

The results for the forecasting problem are shown in table 7. In this case, the MEE norm performed best for the test set although its TI was the smallest of all. The reason for this seemingly contradictory behavior may be better understood if we look at figures 2 and 3. The data, both true and forecasted are displayed. Clearly the high frequency spikes are followed more closely by the BMLP. But these learning accuracy turns out not to be desirable.

Table 7. Performance of Different Training Methods for Forecasting Problem (Genetic and Backpropagation)

TEST (12 ELEMENTS IN THE SET)	
	Average Error
Exponential	5.74%
Mean Absolute	9.34%
Mean Square	10.04%
MiniMax	9.50%
BMLP	17.67%
TRAIN (108 ELEMENTS IN THE SET)	
	Average Error
Exponential	10.62%
Mean Absolute	5.27%
Mean Square	5.26%
MiniMax	5.68%
BMLP	3.35%

Figure 2. Exponential Norm Forecasting.

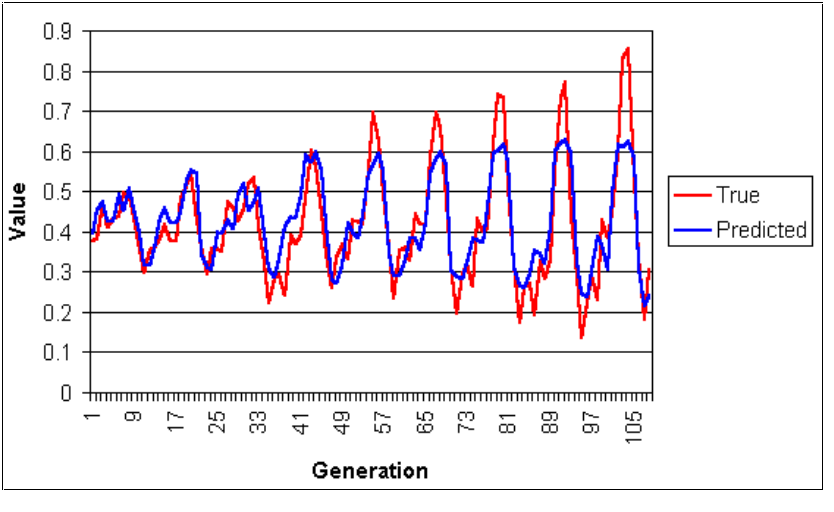
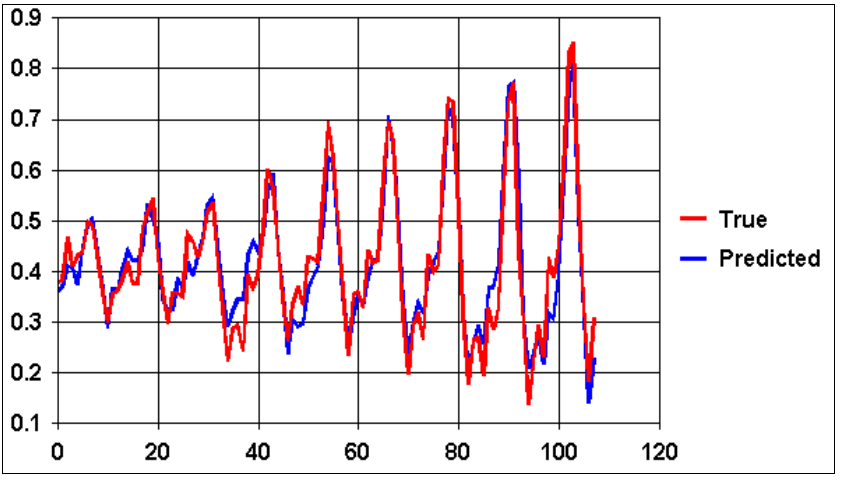
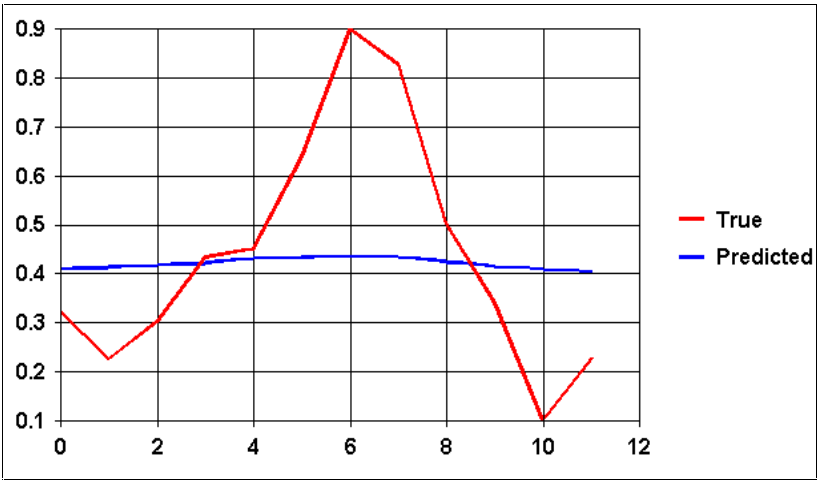


Figure 3. BMLP Forecasting.



The undesirable features of a high TI are clearly reflected when trying to generalize. The distinctly different behaviors of the BMLP and GNN upon generalization are displayed in figures 4 and 5. The predicted values for the case of the BMLP suffer form an accute flattening which importantly affects its generalization index.

Figure 4. BMLP Forecast for Test Set



On the other hand, the MEE normed GNN does not suffer from the mentioned defects. It would be interesting to present the similar behavior of the other GNNs. It is impossible to do so here for reasons of space. Most GNNs, nonetheless, exhibit similar behaviors.

Finally, in figures 6 and 7 we present the training errors for the BMLP and the MEE normed GNN respectively. In figure 6 we show the RMS and Maximum

training error for this problem. In figure 7 we show the MSE for the train error and the test error throughout the process. Notice the difference in scales in these figure which is due to the fact that for the GNN the MSE is an overall error, whereas in the BMLP it is averaged for each set of outputs.

Figure 5. Exponential Forecast for Test Set

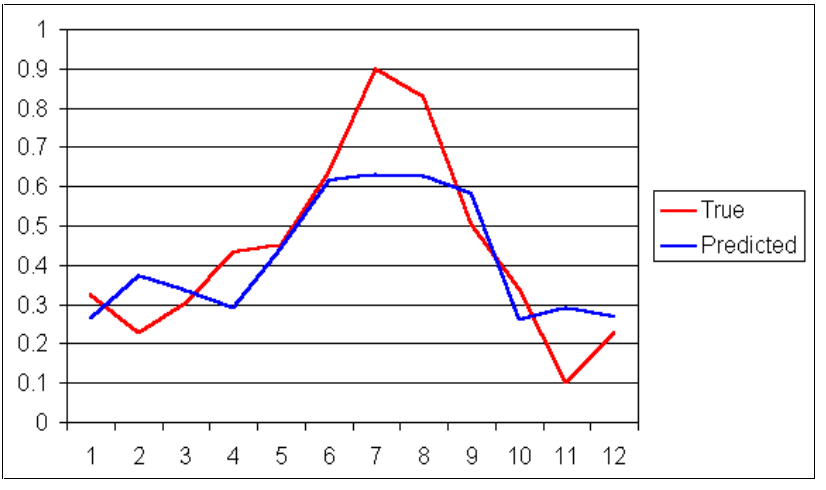


Figure 6. Training Errors for BMLP

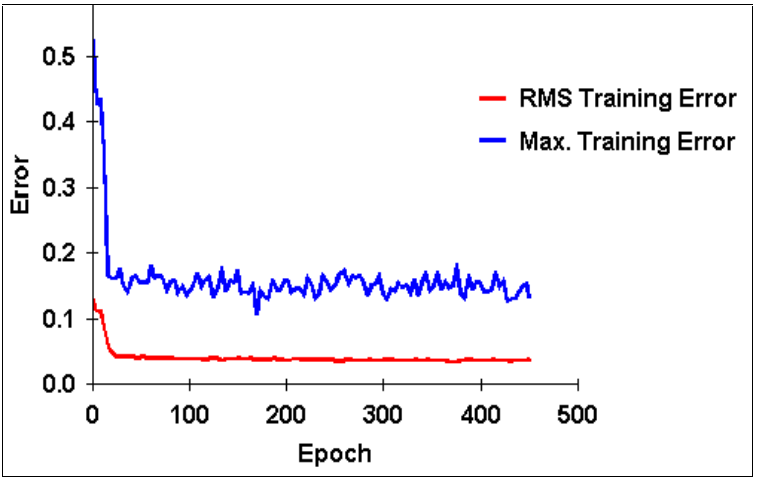
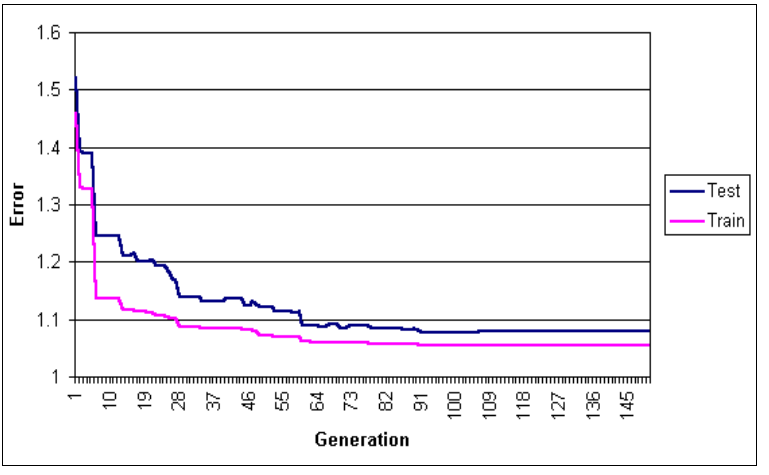


Figure 7. Errors for MEE Norm



4 Conclusions

The conclusions we present here are preliminary. Our first intention is to explore general issues and we, therefore, make no claim to generality. However, several issues have become clear. These we divide in two classes: 1) Conclusions relating to the use of GAs to training NNs, 2) Conclusions relating to the usage of alternative norms *versus* the standard least squares.

4.1 Training NNs with Genetic Algorithms

It is indeed possible to use successfully Gas to train NNs. This result is not surprising nor is it new. Something that should be stressed, however, is that in this work we report a set of applications on practical problems without doing any special adaptation of the GA to the NN problem. This contrasts with various efforts reported in the past where genetic operators were modified to achieve this goal. Of course this is possible because we are using an EGA which is much better than other variations.

In fact, a tacit assumption has been that the EGA will not get stuck in local optima and will continuously increase its precision. This assumption is correct; it is also not the basic issue. Of course a weak GA will hamper our efforts at optimization of the TI. But our main focus is, rather, the GI. The EGA performs adequately for all the tested norms.

Parameter selection is also dispensed with. We have not given an account of the methodology we followed to determine the BMLP parameters. It is only relevant to mention here that such process is not simple and that it is totally unnecessary when using a GA.

In this regard, therefore, we a) Guarantee convergence and b) Disregard parameter selection. The obvious disadvantage is, of course, that GNNs are intrinsecally more costly than their non-evolutionary counterparts.

4.2 Non-standard Norms for NN Training

NNs trained with non-standard norms may perform as well or better than the usual least squares norm (L2) used in backpropagation. Particularly interesting is the fact that different norms perform better than L2 when faced with clustering and forecasting problems. That we may select from a family of norms when facing different problems is interesting in itself. But it is more appealing, from a practical standpoint, if we would had been able to determine a better norm in general. This we are unable to do yet. But from the partial results we have obtained it is possible to speculate that a viable candidate "best" norm is the SAE norm. We have both experimental and theoretical arguments to support this preliminary conclusion.

From the reported set of experiments it turns out that SAE outperforms globally all of the alternative norms (including the BA) when generalizing. It is slightly better than BA in clustering, equally good in logistic classification and quite superior in forecasting.

Theoretically, since SAE tries to minimize the difference $|O(i)-Z(i)|$ for *all* possible output neurons and *all* possible samples it is reasonable to expect a more stringent bound on the problem. At the same time, since the maximum error is minimized, we should expect smaller variances in the trained network's output. This will, intuitively, yield better generalization properties.

4.3 Future Work

We expect to explore the MEE, MAE and SAE norms on a much wider range of problems. Our expected conclusion is that we will be able to derive either a better norm for a wide range of problems or a couple of norms to apply in different sorts of problem.

Another issue we want to explore relates to finding an algorithm which will allow the GNN to stop depending on the performance of the test set. In the reported work, we have simply set limits on the number of runs based on previous knowledge of the problem. But it is reasonable to assume that the algorithm will be able to determine the appropriate point of termination once we understand better the variations of the different norms in time.

Finally, once we have a clearer view of how to set up the GNNs we shall move in the direction which is most interesting and, actually, has motivated this preliminary work: the evolution of NN's topologies.

References

[RUME86] Rumelhart, D.E., Hintoen, G.E., and Williams, R.J., "Learning internal representations by erro propagation", in D.E. Rumelhart, J.L. McClelland, and the PDP Research group, *Parallel Distributed Processing,* Volume I: Foundations, MIT Press, 1986

[MONT89] Montana, D.J., and Davis, L.D., "Training feedforward Networks using Genetic Algorithms", in *Proceedings of the International Joint Conference on Artificial Intelligence*, Morgan Kauffman, 1989.

[MILL89] Miller, G.F., Todd, P.M., and Hedge, S.U., "Designing neural networks using genetic algorithms", in J.D. Schaffer, ed., *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kauffman, 1989.

[KITA90] Kitano, H., "Designing Neural Networks using genetic algorithms with graph generation system", *Complex Systems* 4:461-476, 1990.

[RUDO97] Rudolph, G., "Convergence Analysis of Canonical Genetic Algorithms", IEEE Transactions on Neural Networks, **5**(1):96-101, January, 1997.

[GOLD87] Goldberg, D.E., "Simple Genetic Algorithms and the Minimal Deceptive Problem", in L.D. Davis, ed., *Genetic Algorithms and Simulated Annealing*, Research Notes in Artificial Intelligence, Los Altos, CA, Morgan Kauffman, 1987.

[MITC92] Mitchell, M., Forrest, S., and Holland, J., "The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance", in F.J. Varela and P. Bourgine, eds., *Toward a practice of autonomous systems: Proceedings of the First European Conference on Artificial Life*, MIT Press, 1992.

[HOLL75] Holland, J., "Adaptation in Natural and Artificial Systems", Ann Arbor, Mich., University of Michigan Press, 1975.

[KURI99] Kuri, A., *A Comprehensive Approach to Genetic Algorithms in Optimization and Learning, Theory and Applications, Volume 1: Foundations*, pp. 215-219, Ed. Politécnico, 1999.

[YAOX93] Yao, X., "A Review of Evolutionary Artificial Neural Networks", International Journal of Intelligent Systems, 8:539-567, 1993.

[ROOI96] van Rooij, A.J.F., Jain, J.C., and Johnson, R.P., *Neural Network Training using Genetic Algorithms*, World Scientific, 1996.

Extending the Prediction Horizon in Dynamic Bandwidth Allocation for VBR Video Transport

Armando García-Rodríguez[1], Ramón M. Rodríguez-Dagnino[2], and Christos Douligeris[3]

[1] Lucent Technologies
[2] ITESM, Centro de Electrónica y Telecomunicaciones, Sucursal de correos "J" C.P. 64849, Monterrey, N.L., México rmrodrig@campus.mty.itesm.mx
[3] University of Miami, Electrical and Computer Engineering, Coral Gables, FL 33124 USA, christos@ece.miami.edu, and University of Piraeus, Greece

Abstract. This paper presents a dynamic bandwidth allocation system for real-time variable bit rate (VBR) video transport in asynchronous Transfer mode (ATM) networks. This system takes advantage of scene changes in the video trace on a scale larger than a second, and it adapts the bandwidth as needed. An improvement on efficiency is achieved by assigning bandwidth for the transport of VBR video and having more than one predictor in parallel with different prediction horizons, hence this scheme reduces the processing time for the bandwidth adaptation with no significant degradations on the queue statistics. The link capacity required for a specific session is a function of the input traffic, which is characterized by its spectral characteristic. In particular, we use the low frequency band of the power spectrum, which is extracted from the stochastic input by a low pass filter. The predictor is a neural network (NN) called "Pi-Sigma Network", and the output of this predictor is interpreted according to the prediction horizon in use.

Keywords: Dynamic bandwidth allocation, prediction, neural network, VBR video transport.

1 Introduction

Asynchronous Transfer Mode (ATM) networks provide a flexible manner to manage bandwidth allocation to different classes of services, this feature may be used to give a certain quality of service to each application. The VBR video is an application that may require low cell loss transmission and dynamic bandwidth allocation.

A difficulty that the VBR video traffic transport presents is just the management of the variability in the stochastic traffic trace, which needs to be characterized in order to make a proper bandwidth assignment in a specific time interval, this is an inherent result of constant quality video encoders.

Unfortunately, if we look at the bit rate process as a time series stochastic model, we find that most of the full-motion video traces present bit rate trends

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 365–375, 2000.

that cannot be captured by stationary models. Moreover, the scene boundaries are not well-defined in these video traces. However, an important subclass of these models is the videoconference-type application, where a few main scenes may exist and the scene transitions are well-defined by abrupt bit rate changes. This videoconference-type VBR video traffic has been extensively studied in [1], where the traffic *characterization* is based on linear models of entropy-related index processes, the traffic *modeling* of the trend process is based on semi-Markov stochastic processes with Gamma distributed scene durations, and the scene duration *prediction* is based on nonlinear transformations on the Gamma distributed holding times data, which produce quasi-Gaussian stochastic processes, then the scene duration can be predicted by using well-known results from ARMA time series stochastic modeling. On the other hand, the trend in some video traces has a difficult nature related with the information content on the video sequence. Then, it may not be an easy task to apply the previous videoconference-type of predictors to full-motion video traces. Another approach that several researchers have been used in the literature is to model stochastic trends by quantizing the bit rate levels, and then apply Markov chain models to the discretized bit rate pattern. It seems that by increasing the number of quantization levels we may represent the trend process in a more accurate manner, however, the number of states in the Markov chain will also increase. Additionally, we should mention that this discretization procedure does not guarantee that the resulting Markov chain will be stationary. [1]

In spite of the nonstationary behavior of the time series trend process that appears in some video traces, some approximate results can be obtained by "estimating" the power spectrum of the stochastic video trace by evaluating the power spectrum of the equivalent Markov chain. This approach has been explored in [2] and [3], and the authors have concluded that the low frequency (or trend) component of the video trace is the most influential component for dynamic bandwidth allocation. In this paper, we will follow the previous approach, namely the power spectrum estimation of the equivalent Markov chain.

In order to make good predictions the following tradeoff is stated in [4]: The bandwidth adaptation rate must be slow enough to overcome the adaptation processing time and fast enough to reconstruct the original signal. However, in this paper we show that the bandwidth adaptation rate should not be necessarily unique in order to reconstruct the signal, hence the average adaptation rate can be reduced by taking advantage of intervals where the bit rate remains essentially constant. We found that our proposed scheme, which combines a predictor with a longer horizon for the intervals with almost constant bit rate levels, and a predictor for the fast varying bit rate regions with has a prediction horizon based on the Nyquist rate, results in a more efficient dynamic bandwidth allocation.

In summary, our proposed system is similar to the system in [4], see Figure 1. However, we are adding a variance estimator of filtered input video traces, and the output of this variance estimator gives a measure of the variability on the

[1] A possible way of checking for stationarity in the resulting Markov chain is presented in [1].

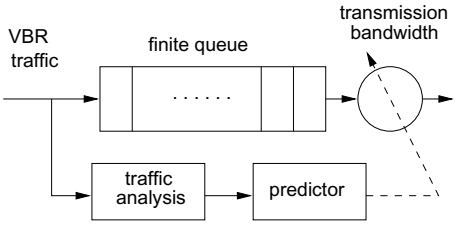


Fig. 1. Dynamic bandwidth allocation system [4].

input process. We use the information provided by the variance estimator to select the appropriate prediction horizon, consequently, the adaptation rate, see Figure 6.

This paper is organized as follows. In Section 2, we review the basic dynamic bandwidth allocation concepts for on-line traffic processing. The predictor is based on a Pi-Sigma neural network described on section 3. In this section we also explain how to extend the prediction horizon according to the variability information of the input process. In Section 4, our proposed system is explained and compared with the dynamic scheme proposed in [4]. This comparison is done by observing the queueing distribution. Finally, our conclusions are presented in Section 5.

2 Bandwidth Allocation

As it has been indicated above, the link capacity needed for an efficient real-time video traffic transport can be evaluated by knowing the low frequency band of the input spectrum. In order to obtain this low frequency component we have implemented an FIR filter of length $N = 1827$ with unitary pass-band gain, and cut-off frequency ω_c. We have used the Kaiser window for its design, 1 Hz cut-off frequency for the passband, 1.7 Hz for the stopband, and ripple in the stopband of 0.02. The magnitude response of this filter is shown in Figure 2.

Now let $x(t)$ be the input traffic, then after filtering we only have the low pass component of the traffic signal, say $x_L(t)$. Let $\mu(t)$ be the link capacity used at time t, then the bandwidth can be assigned as [3]

$$\mu(t) = Cx_L(t) \tag{1}$$

where C is a control parameter that makes $E[\mu(t)] > E[x(t)]$ and then avoids congestion. Provided that $E[x_L(t)] = E[x(t)]$, we can write $E[\mu(t)] = CE[x_L(t)]$ as

$$E[\mu(t)] = CE[x(t)] \tag{2}$$

with $C^{-1} = \rho$ where ρ is the average link utilization.

This scheme allocates the bandwidth dynamically, however, the value of ω_c is not known. A reasonable value may be selected by looking at the power spectrum of the traffic.

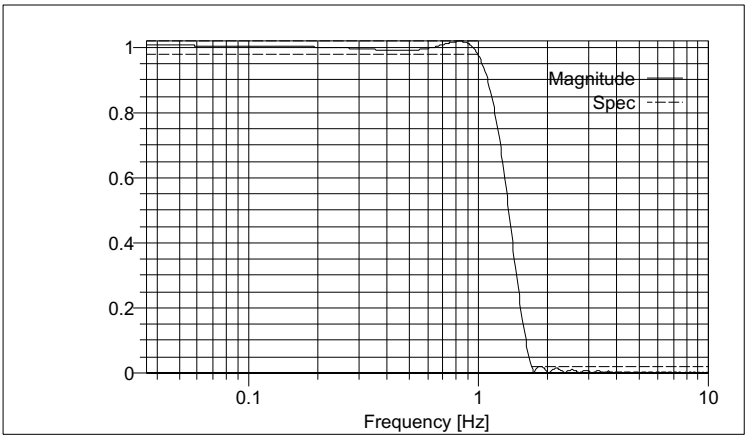


Fig. 2. Magnitude response of the FIR low pass filter.

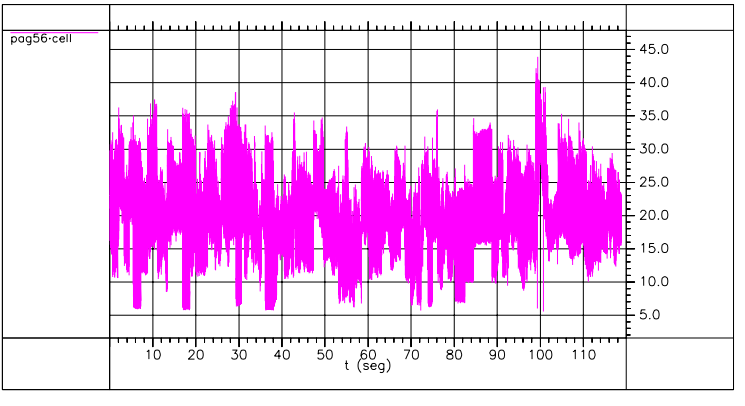


Fig. 3. Input traffic measured in cell/slice, page 56 of Star Wars video trace.

2.1 Traffic Power Spectrum

The traffic source used in this study is the Star Wars video trace released by Bellcore [5]. This movie has been VBR encoded with the following organization: 60 pages of 120 seconds, each second has 24 frames and each frame is formed by 30 slices, then each slice has a period of 1.389 msec. This traffic source exhibits many bit rate transitions as it is shown in Figure 3, and most of its power spectrum is located at a very low frequency band with discrete components at multiples of 24 Hz, which is expected due to the frame to frame correlation, see Figure 4.

After a simple qualitative observation of the input spectrum, a value of 2π rad/sec may be selected for the cut-off frequency ω_c.

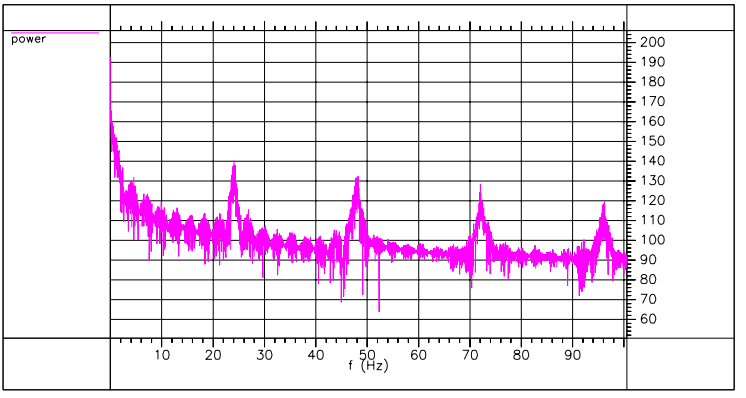


Fig. 4. Power spectrum of the input traffic, page 56 of Star Wars.

3 Bandwidth Prediction

The general topology of the Pi-Sigma neural network is shown in Figure 5 [6]. The first layer, or input layer, copies the input patterns. Define the input vector as $\mathbf{x} = [x_0\ x_1\ \cdots\ x_N]^T$, then each of the input elements x_k is sent to each node of the second layer. We should mention that typically $x_0 = 1$. The second layer, or *sigma* layer, performs K sums of the weighted $(N+1)$ inputs. Define the weight vector $\mathbf{w}_j = [w_{0j}\ w_{1j}\ \cdots\ w_{Nj}]^T$ with $j = 1, 2, \ldots, K$, then

$$h_j = \mathbf{w}_j^T \mathbf{x} = \sum_{k=1}^{N} w_{kj} x_k + w_{0j} \quad ;\ j = 1, 2, \ldots, K \tag{3}$$

where K is the order of the network. The third layer, or *pi* layer, multiplies the K outputs of the sigma layer and applies the sigmoid activation function defined by $\sigma(u) = (1+e^{-u})^{-1}$. It means that the net input to the third layer is given by $\prod_{j=1}^{K} h_j$. The main advantage of this kind of predictor is its ability to represent very complex functions in an efficient manner. Then, each output y_i is obtained as

$$y_i = \sigma_i \left(\prod_{j=1}^{K_i} \sum_{k=0}^{N} w_{ijk} x_k \right) \tag{4}$$

then the vector $\mathbf{y} = [y_1\ y_2\ \cdots\ y_M]$ is the vector of predicted outputs, and w_{ijk} is the weight corresponding to the input x_k at the sigma unit j, and output i.

The inputs on the neural network are sampled versions of $x_L(t)$.

Since $x_L(t)$ is band-limited to the cut-off frequency, the maximum sampling period to handle the signal with no loss of information may be defined by the Nyquist criterion

$$T_s \leq \frac{\pi}{\omega_c} \tag{5}$$

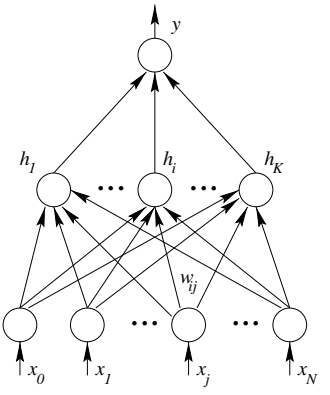


Fig. 5. General topology of Pi-Sigma neural network.

By writing the sampling period of $x_L(t)$ as a factor of the sampling period τ (slice period) of the original signal $x(t)$ with $T_s = \Delta\tau$, we have

$$\Delta \leq \left\lfloor \frac{\pi}{\omega_c \tau} \right\rfloor \tag{6}$$

By considering a sampling period equal to $\tau = 1.389$ msec, and the cut-off frequency $\omega_c = 2\pi$, we obtain $\Delta \leq 360$ from the Nyquist criterion. However, a conservative value $\Delta = 100$ is selected in [4] with the purpose of capturing the maximum of information during abrupt bit rate changes, and then to minimize bandwidth overassignments. In the next section we show that it is possible to choose larger values of Δ, for some intervals of the signal $x_L(n)$, based on the ongoing characteristic of the scene duration rather than the Nyquist criterion.

3.1 Extending the Prediction Horizon

Let the signal $x_L(t)$ be sampled at a Δ_1 period to result in a discrete signal $x_1(n)$, and also sampled at Δ_2 to give $x_2(n)$. We know that $x_L(t)$ is bandwidth limited and assume that the maximum sampling period allowed for reconstruction is Δ_m, say the Nyquist criterion.

Our objective is to perform a prediction of $x_L(t)$ from the reconstruction of one or both of its sampled versions.

If $\Delta_1 < \Delta_2$, then the data represented by $x_2(n)$ is a subset of $x_1(n)$ provided that $\Delta_1 = k\Delta_2$ for some integer k. The starting time for both samplers is the same, i.e., $x_1(0) = x_2(0)$, and in order to keep the system simple, both sampled signals should be synchronized. This requirement has the only purpose of storing less data from the original signal. On the other hand, if $\Delta_1 < \Delta_m < \Delta_2$ then $x_2(n)$ does not provide enough information for reconstruction and the prediction based on this sampled signal is poor.

Note that the bandwidth limitation of $x_L(t)$ may not provide enough information about the slow variation of the signal, just the fast variation. Then the

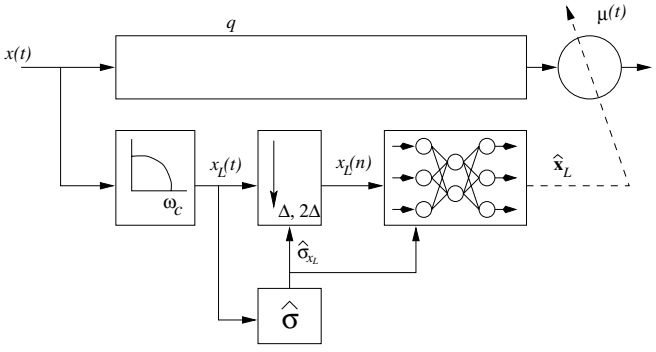


Fig. 6. System to extend the Prediction Horizon in dynamic bandwidth allocation.

$x_L(t)$ signal varies slowly during some time intervals and rapidly during others. Based on this observation, the prediction horizon can be extended by analyzing the variance of the input process, and by selecting the predictors accordingly. The variance is estimated and updated in each minimum sampling period considered.

3.2 Consequences at the Predictor

In essence the neural network works with a static, time-ordered delay line. We should notice that the predictor does not need specific time information, it only needs a carefully specified order relation at the input.

When the sampling period is modified, then the order of the input is not altered, however, there must be a reinterpretation of the specific time information at the output stating that the same difference in time between the inputs should be applied to the predicted outputs.

4 Performance Evaluation

Our proposed system uses two sampling periods after the filter, an estimator of the variance, and the Pi-Sigma neural network as a predictor, see Figure 6. In our simulation, the Pi-Sigma network is trained (supervised training) with 600 patterns from page 56 of Star Wars data, and the performance of the predictor is tested by using page 20. The training patterns are fed into the network through a time delay line, as it is described on [6]. The neural network predictor has $N = 5$ input units, and $M = 4$ output units. We have also implemented a threshold for the variance estimator, i.e., when the value is below the threshold it is assumed that the process varies slowly so a larger sampling period should be used, if it is above the threshold, then the smaller sampling period is selected. The average link utilization is set to 0.8, and two different adaptation intervals or sampling periods were used, $\Delta_1 = 100$ and $\Delta_2 = 200$. The observation window for

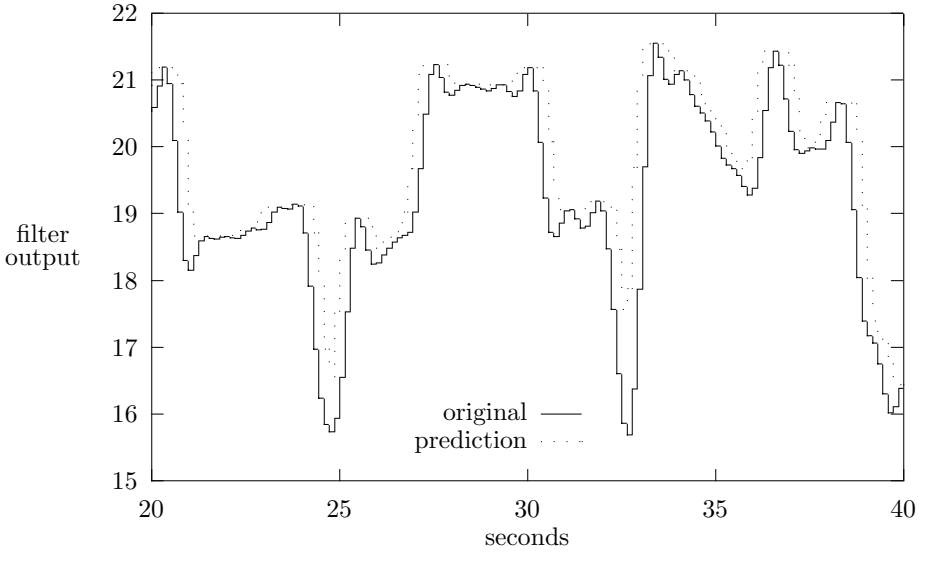


Fig. 7. Prediction on 40 seconds of page 20.

estimating the variance is set at 100 samples. An example of the performance of this predictor is shown in Figure 7.

We consider the following performance measures: the queue distribution, mean length of the queue $\overline{q}$, standard deviation of the length of the queue σ_q, maximum queue length $q_{\max}$ and average link utilization ρ. For the different systems considered in this paper, the number of adaptations are counted in a specific time interval (116.815 sec), and hence the proportion of adaptations under specific sampling periods is calculated.

As a reference, the queue distribution obtained according to the system in [4], is shown in Figure 8. Each point represents the probability that the queue length q exceeds one or more cells than a given value c. Figure 9 shows the queue distribution for our system using four different threshold levels for updating the adaptation rate, namely 0.01, 0.001, 0.0001 and 0.00001. The NN queue distributions on Figure 9 represent the highest probability obtained at each value c by considering the four threshold levels. In the same figure we also show the queue distribution by using a simple predictor scheme that uses the previous sample as the prediction value.

Desirable characteristics of these distributions are that the queue length must be finite and relatively short, then the probability of exceeding the queue capacity is very low.

It can be observed that our proposed system has a very similar queueing distribution performance measure in comparison with the system presented in [4], and it has a better performance than the system with a very simple predictor (previous value predictor).

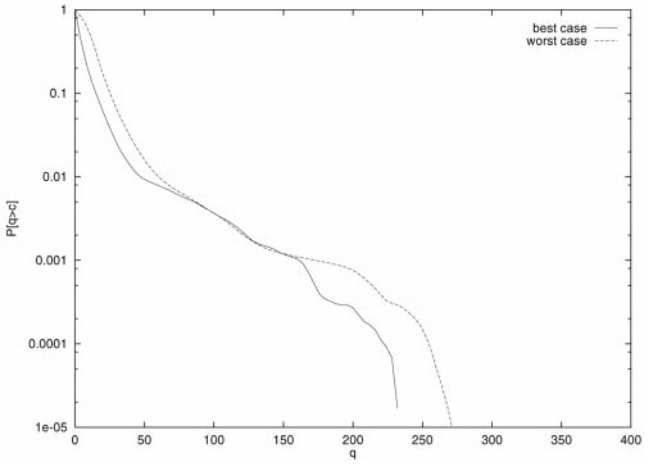


Fig. 8. Queue distribution for a dynamic bandwidth allocation [4].

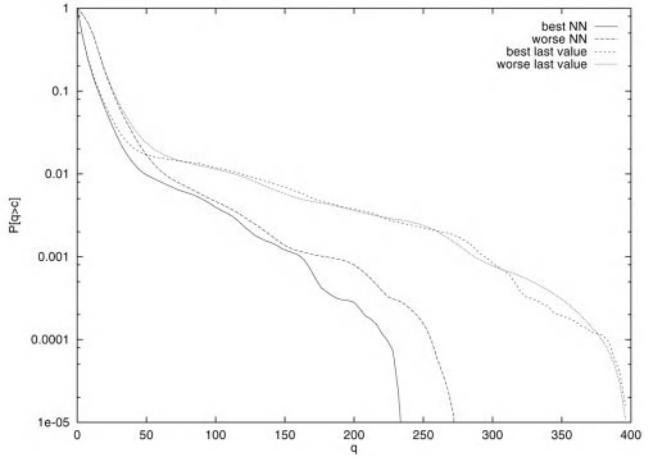


Fig. 9. Queue distribution for the proposed dynamic bandwidth allocation system.

In Table 1 we are basically comparing three systems. The *ideal* system does not consider any prediction scheme and the bandwidth can be adapted instantaneously [3]. The *zero* threshold system which considers a predictor with adaptations made at fixed periods [4]. Finally, our proposed predictor scheme considers four different thresholds for selecting the adaptation rate. As we may see, the performance parameters are very similar, suggesting that our proposed system has a minimum effect on the queue statistics.

The main advantage of our system is shown in Table 2, where the performance measure is the total number of adaptations. For instance, the *ideal* scheme requires $841 \times \Delta_1 = 84,100$ adaptations, it means that the prediction horizon is only

Table 1. Dynamic Allocation and Prediction Horizon Extension, the best case is shown in parenthesis.

threshold	$\overline{q}$	σ_q	$q_{\max}$	ρ
ideal	11.1	13.1	224	0.8
zero	12.954 (6.143)	13.663 (11.885)	277 (234)	0.823 (0.822)
10^{-5}	13.060 (6.141)	14.160 (11.823)	277 (234)	0.823 (0.822)
10^{-4}	13.060 (6.183)	14.172 (12.081)	277 (234)	0.823 (0.822)
10^{-3}	13.060 (6.183)	14.183 (12.071)	279 (234)	0.824 (0.823)
10^{-2}	13.092 (6.193)	14.268 (12.033)	279 (234)	0.824 (0.824)

Table 2. Adaption Bandwidth with Prediction Horizon Extension.

threshold	Δ_1		Δ_2		total
	% updates		% updates		
10^{-5}	97.31	(797)	2.68	(22)	819
10^{-4}	86.35	(639)	13.64	(101)	740
10^{-3}	63.69	(393)	36.30	(224)	617
10^{-2}	31.26	(156)	68.73	(343)	499

one sample, the *zero* scheme requires 841 adaptations in a period of 116.815 sec, in this case the prediction horizon is always $\Delta_1 = 100$ samples, and our system only requires 819 adaptations in the less favorable case (10^{-5} threshold).

This table also shows the proportions of Δ_1 and Δ_2 update periods for different threshold levels for our system. It is evident that this system is really using both adaptation periods and the proportions of each one depends on the threshold level. It can be observed that as the threshold level decreases the system tends to use the shorter adaptation period Δ_1. It means that it is possible to use more than one adaptation period depending on the variability of the traffic.

In summary, while the performance at the queue is maintained, the total number of adaptations can be reduced with no side effects on the queue statistics. This reduction of adaptations is possible because of the presence of intervals with low variability of the filtered traffic, which exists even for full motion video traces such as Star Wars.

5 Conclusion

This paper has presented an improvement on the prediction horizon for a dynamic bandwidth allocation scheme to transport VBR traffic. The extension of the prediction horizon is based on detecting long scene changes, typically longer than one second. The detector for such a variability is a simple variance estimator of the filtered traffic. One major consequence of our scheme is that the efficiency has been improved by reducing the number of adaptations for bandwidth assignment. The system proposed has been compared with other similar systems, and it has been shown that the perfomance is maintained in terms of the queueing statistics.

Acknowledgements. The authors would like to thank the partial support provided by Conacyt (grant C27443-A), NSF (grant ANI-9850134), and Lucent Technologies.

References

1. Ramón M. Rodríguez-Dagnino, *Packet video traffic modeling and prediction in broadband networks*, Ph.D. thesis, University of Toronto, 1993.
2. San qi Li and Chian-Lin Hwang, "Queue response to input correlation functions: Continuous spectral analysis," *IEEE/ACM Transactions on Networking*, vol. 1, no. 6, pp. 678–92, Dec. 1993.
3. San qi Li, Song Chong, and Chian-Lin Hwang, "Link capacity allocation and network control by filtered input rate in high-speed networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 1, pp. 10–25, Feb. 1995.
4. Song Chong, San qi Li, and J. Ghosh, "Predictive dynamic bandwidth allocation for efficient transport of real-time VBR video over ATM," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 1, pp. 12–23, Jan. 1995.
5. Mark Garrett and Martin Vetterli, "ftp://ftp.bellcore.com/pub/vbr.video.trace," *Fourth International Workshop on Packet Video*, aug 1991.
6. Joydeep Ghosh and Yoan Shin, "Efficient higher-order neural network for classification and function approximation," *International Journal of Neural Systems*, vol. 3, no. 4, pp. 323–350, 1992.

Constructing a Query Facility for RBF Networks

Marijke F. Augusteijn[1] and Kelly Shaw[2]

[1] University of Colorado
Colorado Springs, CO, USA
mfa@cs.uccs.edu

[2] Teamshare, Inc. Colorado Springs, CO, USA
kelly.shaw@teamshare.com

Abstract. Artificial neural networks will be more widely accepted as standard engineering tools if their reasoning process can be made less opaque. This paper describes NetQuery, an explanation mechanism that extracts explanations from trained Radial Basis Function (RBF) networks. Standard RBF networks are modified to identify dependencies between the inputs, to be sparsely connected, and to have an easily interpretable output layer. Given these modifications, the network architecture can be used to extract "Why?" and "Why not?" explanations from the network in terms of excitatory and inhibitory inputs and their linear relationships, greatly simplified by a run-time pruning algorithm. These query results are validated by creating an expert system based on the explanations. NetQuery is also able to inform a user about a possible change in category for a given pattern by responding to a "How can I..?" query. This kind of query is extremely useful when analyzing the quality of a pattern set.

1 Introduction

Artificial neural networks (ANNs) have become standard tools for pattern classification problems. However, the knowledge acquired by an ANN during training is generally opaque, and determining why a network has classified a given pattern into a specific class is not an easy task. The capability to explain network behavior is an important one for several reasons. First of all, networks do make errors, and the ability to understand the cause of these errors is often a first step to correcting them. Additionally, ANN learning is different from human learning, and knowledge of what a network has learned may provide additional insight into a problem. Humans also tend to mistrust a technology they do not understand. In order for ANNs to become more widely accepted as standard engineering tools, it will be necessary to associate network processing with explanation generating tools.

Since Hinton has shown that ANNs do indeed learn coherent distributed representations of symbolic concepts [1], many authors have defined algorithms to extract those concepts. These techniques generally involve generating rule sets from a trained network; for example, see [2] – [4]. A smaller number of authors use a network as an oracle for a query mechanism. Network query algorithms provide an ex-

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 376—388, 2000.

planation capability so an ANN user can ask the network why it did, or did not, compute a specific output. Framling [5] describes a mechanism to query a network about contextual importance. Gallant [6] also focuses on extracting explanations from trained networks rather than on extracting rules. He trains a specially designed network architecture and generates "Why?" and "Why not?" queries by examining the activations of the network units. Garson [7] uses trained back-propagation networks to assign relative importance to input values by partitioning the output layer weights based on their strength relative to each other, assuming neurons are active within their linear range. Srivastava [8] computes the relative importance of each input to each output by calculating the importance of each hidden unit to the output and in turn computing the importance of each input to each hidden unit.

This paper introduces a new tool, NetQuery, which generates explanations from a trained network. The architecture is essentially a Radial Basis Function (RBF) network [9]. RBF networks are well suited for explanation generation because they contain a set of locally tuned units. Standard RBF networks are modified for this purpose to identify dependencies between the input values, to be sparsely connected, and to incorporate an easily interpretable output layer. The applications are pattern classification problems. NetQuery extracts "Why?" and "Why not?" explanations from a trained network and uses not just excitatory input values as previous researchers have done, but also inhibitory input values and relationships between input parameters. The RBF axes are associated with fuzzy sets allowing the contributions of individual input units to be described in linguistic rather than mathematical terms. Another new aspect included in NetQuery is the use of run-time pruning, selecting the set of input values minimally necessary for the classification of a specific pattern. This greatly simplifies the explanations associated with the classification. Finally, a new kind of query denoted by "How can I?" is introduced. This kind of query guides a user who wants to change the category of a specific pattern by calculating necessary changes to the input values.

Although the emphasis is on the generation of explanations, not rules, a rule extraction capability is used to validate the "Why?" and "Why not?" responses.

2 The RBF Network Architecture

2.1 Network Criteria

Three aspects of the network architecture will facilitate the extraction of explanations: minimal network size, sparse connectivity, and a shape of the radial basis functions that reflects the pattern distributions. Since concise explanations are preferred, the network should incorporate as few RBF nodes as necessary for correct processing of the data. Given the training data, an optimal clustering algorithm must be designed to partition these data into the smallest possible number of clusters. Furthermore, a network should include only those features relevant to the problem, and individual RBF units should be connected only to those input features relevant to that specific unit. An algorithm must be designed to prune globally and locally useless features. Addi-

tionally, explanations should represent classification relationships as reflected by the data. Therefore, the algorithm determining the direction and size of the RBF axes should calculate these quantities so that the shape of the basis functions reflect that of the pattern clusters.

2.2 The Architecture

The minimal architecture built to support the explanation generation capability has been described elsewhere [10]; a brief discussion is included here. A constructive clustering algorithm is used to find a network with the fewest possible RBF nodes. Clustering is performed in a supervised manner with all patterns grouped into the same cluster belonging to the same category; the RBF centers are now defined.

The radial basis functions themselves are chosen to be elliptical Gaussians. The direction of the axes of these (hyper)-ellipsoids must be aligned with the principal components of the associated cluster. Given the patterns of a cluster, first calculate their covariance matrix. Each eigenvector of this matrix defines a principal component direction and therefore serves as the direction of an ellipsoid axis; the associated eigenvalue determines the axis' width. Since we do not assume that the various clusters are aligned, each ellipsoid will have a different set of directions. Thus, each RBF is associated with a specific rotation matrix $\boldsymbol{P}$ resulting in an additional layer, called the rotation layer, in the network architecture. The input and rotation layers are completely connected with the weight values set equal to the rotation coefficients. Each rotation unit is only connected to its associated RBF. These connections represent the axes of the RBF and are weighted by two values: a center and width component. Figure 1 shows this architecture.

Given the k^{th} RBF unit with center μ_k and width σ_k, and given input pattern $\boldsymbol{X}_0$, the RBF output value is calculated as:

$$RBF_k(\mathbf{X_0}) = \exp\left[-\sum_{i=1}^{n} \frac{\{(\mathbf{P_k X_0})_i - \mu_{ki}\}^2}{\sigma_{ki}^2}\right] \tag{1}$$

The weights from the RBF layer to the output units are each set to one of three values from the set {-1, 0, 1}. These weights are interpreted as inhibitory (-1), excitatory (+1) or pruned (0). No training is required to determine these output weight values; they are set following the rule that an RBF unit associated with class *j* should excite the output unit of this class and inhibit all others. However, the output layer also has bias weights; the value of the bias of the k^{th} output unit is calculated as:

$$w_{k0} = \frac{\sum_{i=1}^{nTrainingPairs} \left(d_{ik} - \sum_{j=1}^{nRBF} w_{kj} RBF_j(\mathbf{X_i}) \right)}{nTrainingPairs} \tag{2}$$

in which *nTrainingPairs* is the number of patterns in the training set, d_{ik} is the target for node *k* given input pattern $\boldsymbol{X}_i$, *nRBF* is the number of RBF units, and RBF_j is the

output of the j^{th} RBF given input $\boldsymbol{X}_i$. The number of output units equals the number of classes in the problem, and the activation functions of these units are linear.

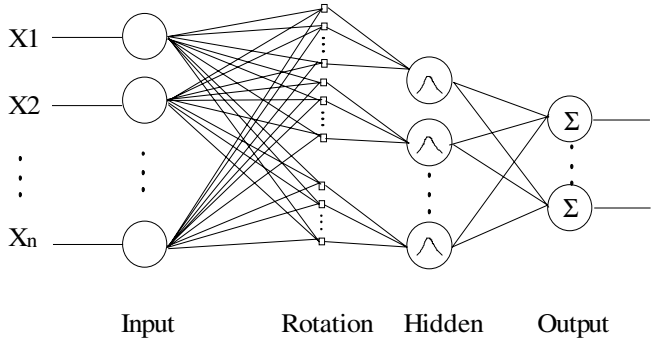


Fig. 1. Structure of the modified RBF network

2.3 Network Pruning

The network must now be pruned to satisfy the sparse connectivity requirement. First, the input layer is pruned to remove those units that contribute little to the classification based on an entropy measure. Next, the RBF axes are considered for pruning using a saliency measure similar to the one introduced in Le Cun *et al*. [11]. It may happen that all axes of some RBF unit are deleted, then the unit itself is also removed. Finally, the rotation and output layers are pruned by removing those connections associated with small weight values. The criterion used in each pruning action is that network performance as measured over a validation set should not decrease. It was found that the pruning algorithm was able to remove a large percentage of the original network connections in our test problems.

3 Test Cases

Pruned networks were built for a variety of test cases; three of them are used here to illustrate the explanation capability. The relatively simple "rule plus exception" problem [12] and iris classification problem are used to show proof of concept. A more complex problem is the Wisconsin breast cancer study. The data for the iris classification and the Wisconsin breast cancer study were obtained from the University of California, Irvine machine learning database repository.

In the "rule plus exception" problem, a network must learn the Boolean function $AB + \overline{A}\overline{B}\overline{C}\overline{D}$, in which the first term defines the rule, and the second term constitutes the exception. The network was constructed using two RBF units and two output units representing the two possible values of the Boolean function. The pruning algorithm removed 71% of the connections, and the pruned network (shown in Figure 2a) was found to correctly classify all 16 patterns.

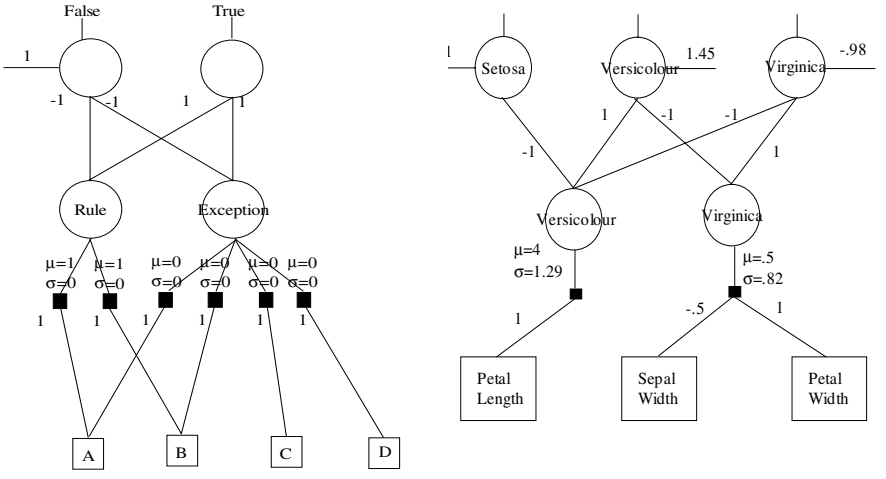


Fig. 2. Pruned networks for the Rule Plus Exception (a) and iris classifiication (b) problems

The iris classification identifies an iris as one of three types based on the four attributes: sepal length, sepal width, petal length, and petal width. The data set has 150 patterns, 50 for each class. A classification rate of 98% was achieved and maintained after 82% of the connections were pruned. Pruning also removed the 'sepal length' attribute from the input layer and the RBF unit corresponding to the Setosa class from the hidden layer. The pruned network is shown in Figure 2b.

The breast cancer data consist of 966 patterns, each having 9 attributes (clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, simple epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitosis) to determine if a woman has benign or malignant tumors in her breasts. A classification rate of 96% was achieved and maintained after pruning. The pruning algorithm removed the uniformity of cell size and mitoses input units and 76% of the weights. The pruned network has two output units (benign and malignant class) and two RBF units and is not shown due to its complexity.

4. Interpretation of the Network Layers

Assume a network has been trained for a specific classification problem. Each node in the RBF layer defines a set of localized requirements for class membership. This local information must first be translated into a set of fuzzy membership functions that will specify, in a symbolically meaningful way, the class membership information encapsulated within the RBF network parameters. Our approach is reminiscent to the work of Dickerson and Kosko [13] who have used rotated ellipsoids to define fuzzy membership functions that are then projected back into the input space. Thus, their method generates rules in the input feature space, not in the rotated space of the ellipsoids. This may seem a natural approach, but it ignores possible linear relation-

ships between input parameters. Our method generates fuzzy relationships in the rotated space and can therefore preserve important linear relationships.

An additional drawback of most algorithms generating fuzzy memberships is that they define a single membership function per RBF axis. These functions describe the extent to which an input is within the validity area of the corresponding axis, but some applications may require more specific information. Suppose that some rule applies only if a specific input is very close to the RBF center. This requires a numeric value (representing the concept "very close") to be associated with the membership function. However, it would be advantageous if the user need not be burdened with these numeric distance interpretations. The alternative is to define a set of membership functions and map all of them onto the associated RBF axis. Table 1 shows the six measures, defined in this work, in terms of the distance between the rotated input pattern component $(PX)_i$ and the center component μ_i in relation to the width component σ_i of a specific RBF axis.

Table 1. Definitions of the membership functions

Membership Term	Definition
VERY CLOSE to	$\lvert\mu_i - (PX)_i\rvert \le .25\sigma_i$
CLOSE to	$.25\sigma_i < \lvert\mu_i - (PX)_i\rvert \le .5\sigma_i$
SOMEWHAT CLOSE to	$.5\sigma_i < \lvert\mu_i - (PX)_i\rvert \le .75\sigma_i$
SOMEWHAT FAR from	$.75\sigma_i < \lvert\mu_i - (PX)_i\rvert \le \sigma_i$
FAR from	$\sigma_i < \lvert\mu_i - (PX)_i\rvert \le 1.5\sigma_i$
VERY FAR from	$1.5\sigma_i < \lvert\mu_i - (PX)_i\rvert$

In order to interpret the output layer, recall that the associated weights can only take on one of three values from the set {-1, 0, 1}. A +1 value means that the connection from the RBF unit to the output node is excitatory; likewise, a -1 value corresponds to an inhibitory connection and a 0 value indicates that the associated RBF is irrelevant to that specific classification. RBF nodes tend to have either high or low output values since Gaussian functions drop off steeply outside their activation fields. The four meaningful combinations of RBF output values and associated connection strengths are summarized in Table 2.

An RBF output value must also be related to the value of each axis, as defined by the equation

$$net_i = \frac{\left\|(\mathbf{PX})_i - \mu_i\right\|^2}{\sigma_i^2} \tag{3}$$

It is seen from Eq. (3) that a large axis value means that the rotated input component is far from the RBF center, and this axis will therefore contribute little to RBF activation. Similarly, low axis values cause high RBF activations. Table 2 also shows the appropriate axis values for the listed RBF activations. Thus, Table 2 indicates how output node results can be propagated back to individual values of the RBF axes.

Table 2. Interpretation of RBF node activation

RBF Activation	Axis Net Value	Hidden-Output Weight	Output Node Result
high	low	1	Excitatory enabled
high	low	-1	Inhibitory enabled
low	high	1	Excitatory disabled
low	high	-1	Inhibitory disabled

5. Query Answering

The aim of this work is to allow a user to query a trained RBF network about specific classification decisions. The most obvious queries are "Why?" (Why was pattern $\boldsymbol{X}_0$ put into class C?) and "Why not?" (Why was pattern $\boldsymbol{X}_0$ not put into class B?). The user may also want to know how some input pattern can be changed from one class to another given some hypothetical constraints. The corresponding questions are called "How can I...?" queries. (How can I move pattern $\boldsymbol{X}_0$ from class C to class B by modifying the value of pattern component X_i?). A software tool called NetQuery was developed that allows a user to load a previously trained network and perform these kind of queries.

5.1 The "Why?" Query

The answer to a *"Why?"* query provides information about why the network placed a specific input pattern into a specific class. NetQuery formulates the explanation in terms of input parameters and values important to the specific classification of that pattern. This explanation also reveals dependencies between the input parameters.

The output layer of the RBF network classifies an input pattern based on a winner-take-all strategy: the output unit showing the highest activation value decides the class of the input pattern. Thus, given the fact that a certain pattern is put into class C by the network, a "Why?" query is answered if NetQuery can relate a high activation of the output unit of class C to the input variables causing this activation. The situation can be analyzed using the combinations listed in Table 2. Obviously, an activated RBF unit with excitatory connection (excitatory enabled) will make a positive contribution to the value of the output unit, while an activated unit with inhibitory connection (inhibitory enabled) will decrease this value. Low RBF activation combined with an excitatory weight (excitatory disabled) does not contribute to the activation of the output unit and need not be taken into account. However, if a low activation of an RBF unit is combined with an inhibitory weight (inhibitory disabled) then it must be regarded as a negative contribution to the output value; that is, the activation of the output unit is high *because* the RBF unit activation is *not* high. Thus, two entries in Table 2 are important when answering a "Why?" query: excitatory enabled and inhibitory disabled.

Consider the "Rule Plus Exception" network as an example. A trained network will classify the input pattern (0 0 1 0) as False. As illustrated by Figure 2, the Rule

RBF node will have a low output value in this case and is connected to the False output unit through an inhibitory weight; it is therefore a disabled inhibitory input to this output layer unit. The Exception RBF also has a low output value and is likewise connected to the False output neuron through an inhibitory weight. In terms of the RBF activations, the resulting explanation then becomes "Input (0 0 1 0) is False because Rule node is low and Exception node is low."

NetQuery then uses the fuzzy membership functions to relate these RBF activations to the given pattern. Any RBF axis having a high net value joined through an inhibitory connection, or any RBF axis having a low net value joined through an excitatory connection can be used to explain a "Why?" query. In the previous example, NetQuery answers "Input (0 0 1 0) is False because C is VERY FAR from 0, A is VERY FAR from 1 and B is VERY FAR from 1." This is a good answer since it explains exactly why the input is False without referring to RBF axis values that do not contribute to the classification of the input. For example, the same RBF node containing the C = 0 axis also contains an A = 0 axis, a B = 0 axis and a D = 0 axis. Since these axes do not have a high net value, they do not contribute to a low RBF activation and are not mentioned in the explanation.

In some larger networks containing many RBF nodes, the contributions from the deactivated inhibitory connections may clutter and obscure the explanations. NetQuery allows the user to turn inhibitory explanations off and report only the disabled interference classes.

The network construction algorithm builds an architecture with as few units as possible and with RBF nodes whose shapes correctly reflect the underlying training data distributions. Every RBF axis has a set of rotation coefficients defining the linear relationships necessary for an input to fall within the receptive field of the axis. However, not all these rotation coefficients may be necessary to classify *individual* test patterns presented to the network, although all coefficients are necessary to maintain the classification rate over *all* training and validation patterns. NetQuery gives the user the option to perform run-time pruning on the network's rotation coefficients. As with pruning during the training process, the contribution of an individual rotation coefficient is proportional to its absolute value. NetQuery can perform run-time pruning by setting small rotation coefficients to zero for disabled inhibitory RBF axes and enabled excitatory axes.

5.2 The "Why Not?" Query

The user may want to know why the network did not place a certain pattern $\boldsymbol{X}_p$ into a specific class. The "Why not?" query may help users identify potential measurement errors in their training data or explain perceived misclassifications. For example, in the breast cancer diagnosis problem, one of the training patterns, misclassified as malignant, cites clump thickness, normal nucleoli and bare nuclei (and their corresponding linear relationships) as offending input values. This information may cause a user to reconsider these three parameters to see if a measurement error has caused the misclassification.

A "Why not?" query is answered if NetQuery can explain why the output unit associated with the class in question has a low activation. Of the four combinations listed in Table 2, two have been used to explain high output unit activations. Of the

remaining two, an enabled inhibitory input directly contributes to a low output unit value, while a disabled excitatory input contributes negatively; that is, the output *would have been high* if this input had been enabled rather than disabled. Consider the "Rule Plus Exception" problem again as an example. When asked why the pattern (0 0 1 0) is not True, NetQuery would answer that the Rule RBF has a low activation as does the Exception RBF. These are the two disabled excitatory inputs to the True output unit. Following back through the output layer connections to the RBF axes, NetQuery would answer "Input (0 0 1 0) is not True because A is VERY FAR from 1, B is VERY FAR from 1, and C is VERY FAR from 0." As with the "Why?" query there are three RBF axes not included in this explanation because they do not correspond to low values of excitatory inputs or high values of inhibitory inputs.

Run-time pruning applies just as well to "Why not?" queries as it does to "Why?" queries. During a "Why not?" query the run-time pruning algorithm is applied to disabled excitatory and enabled inhibitory RBF axis rotation coefficients. The breast cancer diagnosis problem provides an example of the value of run-time pruning for "Why not?" queries. Given the pattern:

Clump Thickness 0.8, Uniformity of Cell Shape 1.0, Marginal Adhesion 0.8, Epithelial Cell Size 0.7, Bare Nuclei 1.0, Bland Chromatin 0.9, Normal Nucleoli 0.7

(classified correctly as malignant by the network), the query "Why not benign?" with no run-time pruning takes the form:

(Clump Thickness) + .3(Uniformity of Cell Shape) + .3(Bare Nucleoli) + .4(Normal Nucleoli) is VERY FAR from 0.0
(Uniformity of Cell Shape) is VERY FAR from .14
(Clump Thickness) is SOMEWHAT CLOSE to .3

When run-time pruning is enabled, NetQuery removes many unnecessary rotation coefficients and generates the explanation:
Noting that two parts of the explanation now refer to the value of "Clump Thickness",

(Clump Thickness) is SOMEWHAT FAR from 0.0
(Uniformity of Cell Shape) is VERY FAR from .14
(Clump Thickness) is SOMEWHAT CLOSE to .3

a postprocessor may be used to remove the less important one.

5.3 The "How Can I…?" Query

A user may want to ask how a specific pattern can be moved from one class to another by changing some of its parameter values. Three different types of queries have been implemented, but only two of them are discussed. The first one addresses the possibility of moving a pattern from one class to another by changing the value of a single component. The second one attempts to accomplish this change by modifying the fewest possible number of component values, and the last one (not discussed in this paper) tries to move the pattern over the smallest distance to another class.

The first query may take the following form: "How can the current input pattern be moved from class i to class j by changing input k?" This query addresses the question if and how two classes may be distinguished by considering a single input compo-

nent. For example, in the iris classification problem, the user may ask how a pattern classified as Setosa can be changed to Versicolour simply by changing the value of petal width. This kind of query can help a user find training data errors. Given some misclassified pattern in the training or test set, the user could ask how much some suspicious component would have to be changed to classify this pattern into the correct class. If the value need only be changed by a small amount then the corresponding measurement may have been correct, but the pattern may be misclassified because it is near the boundary of two categories. If, on the other hand, the value must be changed by a larger amount then either this value may be incorrect, and the user should validate the measurement, or the expected classification may be wrong, and the user must re-evaluate how this pattern is classified.

If the pattern component ranges over a finite set of values then the implementation of this capability is straightforward: NetQuery attempts all values to find the ones that cause the desired change. If more than one value is found NetQuery will select the one closest to the current value, and if no value exists the user will be informed that the desired change is not possible using the selected component. The Breast Cancer Diagnosis problem provides a good example, since its scaled parameters range from 0.1 to 1.0 with 0.1 increments. The following pattern:

Clump Thickness 0.1, Uniformity of Cell Shape 0.1, Marginal Adhesion 0.1, Single Epithelial Cell Size 1.0, Bare Nuclei 0.1, Bland Chromatin 0.1, Normal Nucleoli 0.1

is misclassified by the network as malignant. NetQuery is asked to change the input to benign by modifying the normal nucleoli component and responds: "Move normal nucleoli to 0.4". If, on the other hand, NetQuery is asked to accomplish the same change by modifying bland chromatin it indicates that this is not possible.

If the value of an input component is real then an exhaustive search is not possible. Given pattern $\boldsymbol{X}$ and x_j as the component under consideration then a change Δx_j must be calculated such that the new value $x_j + \Delta x_j$ is within the receptive field of an RBF of the target class. Suppose an appropriate RBF with rotation matrix $\boldsymbol{P}$, center μ, and width σ has been identified. In order for the i^{th} axis of this RBF to contribute positively to its output value, the i^{th} component of the rotated input pattern must be close to the i^{th} component of the RBF center. If:

$$\Delta x_j = \frac{\mu_i - (\mathbf{PX})_i}{P_{ij}} \tag{4}$$

then the modified pattern component will be at the center of the i^{th} axis. NetQuery will perform this calculation for every axis of each RBF of the target class and will report the smallest acceptable Δx_j, if one exists. Take the iris classification problem as an example; here, all components are real valued measurements of iris characteristics. The pattern (petal length, sepal width, petal width) = (4.9, 2.5, 1.5) cm. of the class Versicolour is misclassified by the network as Virginica. When asked how to change this pattern to Versicolour by modifying petal length, NetQuery replied: "Move petal length closer to 4.0".

In some cases it may be more efficient or even necessary to move an input *away from* the center of an RBF axis in the current class rather than *toward* the center of an axis in the target class. When the target class has no corresponding RBF node, for example, no appropriate RBF exists, and the calculation of Eq. (4) cannot be performed. Let $\mathbf{P}_i$ be the rotation vector for an RBF axis in the current class with center μ_i and width σ_i. If the value of Δx_j is such that the inequality

$$\left(\frac{(PX)_i + P_{ij}\Delta x_j - \mu_i}{2\sigma_i} \right) \geq 1 \tag{5}$$

holds, then the input will be moved at least $2\sigma_i$ away from the center of the RBF axis. NetQuery will calculate Δx_j for each RBF axis belonging to the current class using strict equality in Eq. (5).

If any of the Δx_j values that move the pattern away from an RBF center do indeed change its classification to the target class, and the distance is less than any other acceptable Δx_j, NetQuery will tell the user to "Move x_j away from <current value>" rather than "Move x_j toward <new value>." This implies the input can be moved in either direction given the input parameter min/max constraints. Using the same misclassified iris classification pattern: (petal length, sepal width, petal width) = (4.9, 2.5, 1.5) cm., a user may ask how to change its class from Virginica to Versicolour by modifying only sepal width. NetQuery will reply: "Move sepal width away from 2.5." Similarly, when asked how to change to Versicolour by moving petal width, NetQuery will answer: "Move petal width away from 1.5."

It may happen that the required change Δx_j will force $x_j+\Delta x_j$ out of its specified range. In this case, NetQuery will move the value to the nearest range boundary and will then determine the resulting classification. The algorithm will exhaustively examine all appropriate RBF axes to find the one which re-classifies the input vector correctly while moving the input over the smallest distance. The iris classification pattern: (petal length, sepal width, petal width) = (1.4, 3.5, 0.2) cm., correctly classified as Setosa, provides an example. When trying to change the classification to Virginica by changing only sepal width, the calculation results in a value -.6 for this parameter. However, sepal width is constraint to positive values, as a result NetQuery will recommend movement toward 0 instead. A very small value for sepal width will indeed lead to the requested (incorrect) classification in this example. A limiting range may sometimes prevent a requested change altogether.

It will not always be possible to accomplish a requested change in classification by modifying the single, user-specified pattern component. The next query type attempts to find the smallest number of parameters (including a single parameter) and their required changes so that a user-specified classification change will take place, if possible. First, consider the case in which a single component can still accomplish the requested change, but NetQuery has to determine which one. The algorithm, stated in the previous section, can still be used, but must be applied to each component in turn. It then chooses the smallest possible change.

Returning to the iris classification problem: (petal length, sepal width, petal width) = (4.9, 2.5, 1.5) cm., the user may ask how to modify its classification from Virginica

to Versicolour without stating which component to change. NetQuery will find that this change can be accomplished by moving sepal width away from 2.5.

6. Query Verification through Rule Extraction

NetQuery generates responses to a variety of queries, but the question remains if the responses to the "Why?" and "Why not?" queries truly represent the information stored in the network. The query responses must be validated with respect to the network. The "How can I..?" type queries do not require validation, since the responses are distance calculations that are known to lead to the desired change in classification.

The software tool, developed to generate the rule set, is called GenRules. Given a trained network and a training data set, GenRules will run the "Why?" query on every training pattern using the output from NetQuery as a template for rule construction. Details of the rule generation will not be given in this paper. The important fact is that every correctly classified pattern in the training set is used, and that rules are generated from the explanations instead of directly from the trained network. Therefore, if the rules can classify the test set with the same accuracy as the network then the explanations are assumed to contain the same information as the trained network, but in a conceptual instead of numerical format. Before the rules can be used as a classification tool, they must be embedded in an expert system shell. CLIPS, a public domain expert system tool developed by NASA in 1986, was used for this purpose. Table 4 shows the comparison in classification rate between the RBF network and the expert system. This Table clearly indicates that NetQuery is able to generate valid explanations for the "Why?" and "Why not?" queries in the reported test cases.

Table 3. Neural network and expert system performance comparison

Dataset	Network Classification Rate	Expert System Classification Rate
Rule Plus Exception	100	100
Iris Classification	98.0	98.0
Breast Cancer Diagnosis	98.2	97.7

7. Conclusions

Most existing explanation generation methods suffer from one or more of the following problems. They may generate explanations only in terms of independent pattern components, ignoring any interdependency between them. They may explain classification decisions based only on excitatory input values, omitting the contributions of the inhibitory values. They attempt to explain network classifications, but do not support queries leading to changes in classification of a given pattern. The main goal of our research was to create a query facility that could overcome these limitations.

Like many other available techniques, NetQuery is restricted to a certain kind of architecture. The RBF architecture was chosen because pattern information is stored locally, and only a small part of the network must be considered for each query. Sev-

eral modifications were deemed beneficial. The rotation layer was added because meaningful explanations could only be extracted if the activation of each RBF unit reflected the underlying pattern organization. Network pruning was considered essential to avoid the overwhelming amount of data that might otherwise be generated.

This research indicated that excitatory and inhibitory connections are equally important when generating explanations. The tri-polar output weights allowed the integration of both types of connections in a natural way. Linear relationships were also found to be important in some application areas. The classification responses were validated by means of a rule-based system. Queries concerning a change of class require the calculation of a difference vector that, when added to the original pattern vector moved the pattern to within the activation region of an RBF of the desired class. This kind of query can flag suspicious or erroneous patterns and, in addition, indicate which feature components are most likely to be in error.

References

[1] G. E. Hinton, “Learning Distributed Representations of Concepts,” *in Proc. Annual Conference Cognitive Science Society*, p. 1, 1986.

[2] S. B. Thrun, “Extracting Rules From Artificial Neural Networks With Distributed Representation,” in G. Tesauro, D. S. Touretzky and T. K. Leen, Eds, *Advances in Neural Information Processing Systems* 7. San Mateo, CA: Morgan Kaufmann, 1996 pp. 505 - 512.

[3] L. Fu, "Rule Learning by Searching on Adapted Nets," *Proc. 9th National Conference on Artificial Intelligence*, 1991 p. 590.

[4] M. W. Craven and J. W. Shavlik, “Extracting Tree-Structured Representations of trained networks”, in D. S. Touretzky, M. C. Mozer and M. E. Hasselmo, Eds. *Advances in Neural Information Processing Systems* 8. pp. 24 - 30, Cambridge: MA: MIT Press.

[5] K. Framling, "Explaining Results of Neural Networks by Contextual Importance and Utility," *Proc. Rule Extraction From Trained Artificial Neural Networks Workshop*, Brighton, UK: Univ. of Sussex, 1996, p. 43.

[6] S. I. Gallant, “Connectionist Expert Systems,” *Communications of the ACM*, vol. 31, pp. 152 - 169, 1988.

[7] D. G. Garson, “Interpreting Neural-Network Connection Weights,” *AI Expert*, pp. 47 - 51, 1991.

[8] R. P. Srivastava, "Automating Judgmental Decisions Using Neural Networks: A Model for Processing Business Loan Applications," Proc. 1992 ACM Computer Science Conference, 1992, p. 351.

[9] J. Moody, and C. Darken, “Fast learning in a network of locally-tuned processing units,” *Neural Computation* vol. 1, pp 281 - 294, 1989.

[10] M. F. Augusteijn, and K. A. Shaw, “Radical Pruning: A Method to Construct Skeleton Radial Basis Function Networks,” *Int. J. Neural Systems*, under review.

[11] Y. Le Cun, J. Denker, and S. Solla, "Optimal Brain Damage," in D. S. Touretzky, Ed., *Advances in Neural Information Processing Systems 2*. 1990, pp. 598 - 605.

[12] M. C. Mozer, and P. Smolensky, "Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment," in D. S. Touretzky, Ed., *Advances in Neural Information Processing Systems 1*, 1989, pp.107 - 115.

[13] J. Dickerson, and B. Kosko, "Fuzzy Function Learning with Covariance Ellipsoids," *IEEE Int. Conf. Neural Networks,* vol. 3, 1993 p. 1162.

A Procedure to Select the Vigilance Threshold for the ART2 for Supervised and Unsupervised Training

P. Rayón Villela and J. H. Sossa Azuela

Centro de Investigación en Computación-IPN
U. Prof. Adolfo López Mateos, México, D.F. 07738, México
Fax: (52 5) 586 29 36
{prayon,hsossa}@cic.ipn.mx

Abstract. The Adaptive Resonance Theory ART2 [1] is used as a non supervised tool to generate clusters. The clusters generated by an ART2 Neural Network (ART2 NN), depend on a vigilance threshold (ρ). If ρ is near to zero, then a lot of clusters will be generated; if ρ is greater then more clusters will be generated. To get a good performance, this ρ has to be suitable selected for each problem. Until now, no technique had been proposed to automatically select a proper ρ for a specific problem. In this paper we present a first way to automatically obtain the value of ρ, we also illustrate how it can be used in supervised and unsupervised learning. The goal to select a suitable threshold is to reach a better performance at the moment of classification. To improve classification, we also propose to use a set of feature vectors instead of only one to describe the objects. We present some results in the case of character recognition.

1 Introduction

Pattern classification involves mapping a pattern correctly from the so called feature space into a class-membership space [2]. The decision-making process in pattern classification can be thus summarized as follows. Consider a pattern represented as a $n-$dimensional feature vector:

$$X = [x_1, x_2, ..., x_n] \tag{1}$$

The task is to assign this vector to one of the K classes $C_1, C_2, ..., C_K$.

Supervised and unsupervised classification need a description of each object in terms of a feature vector. Supervised learning uses feature vectors with class labels, unsupervised learning uses feature vectors but without class labels. The existence of a specific desired output for each of the training set vectors is what differentiates supervised from unsupervised learning [2].

The feed forward multilayer perceptron is one of the most common neural networks with supervised learning, and is trained for backpropagation [3]. ALOPEX is a stochastic parallel algorithm which treats the learning process in

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 389–400, 2000.

a neural network as an optimization problem [4]. The ART2 NN is an unsupervised classifier that accepts input vectors that are next classified according to the stored pattern they most resemble. The ART2 NN can adaptively create a new cluster corresponding to an input pattern if it is determined to be "sufficiently" different from the existing clusters, this is controlled by a vigilance threshold ρ [5]. Thus the ART2 NN allows the user to control the degree of similarity patterns placed in the same cluster. Until now, no technique had been proposed to select a proper value for this vigilance threshold.

In this paper, we present a way to automatically select the vigilance threshold for unsupervised classification using the ART2, we also suggest how to use the ART2 in supervised classification. We have not found any related reference until now.

It is common in supervised and unsupervised classification to use one feature vector to describe each pattern to be further classified. In this paper we propose to use a set of features vectors to describe a pattern, instead of one. This, as we will see in the following section, will enhance the performance of the classifier.

2 Extracting Sub-feature Vectors

Pattern recognition needs that the object to be recognized is represented in terms of a feature vector. The aim is to have a unique representation for each class and to avoid overlapping among classes (Figure 1), minimizing at the same time missclassification. This is only valid if the object is represented in a feature vector. Nevertheless, if a set of features vectors is used to describe one pattern, then patterns from different classes can share common features.

Considering a distance metric as a similarity measure, the aim is to increase the intraclass distance and to decrease the interclass distance. The most common used similarity measure is the euclidean distance [6]. If this distance is used, in some cases variations among the different elements of the feature vector are not the same, and only some elements will be considered when the distance is computed. Instead of using only one feature vector, we propose to use a set of sub-feature vectors (SFVs) to describe an object. Thus we will have a set of descriptions of the objects with the same variations meaning this that the objects share common features (Figure 2).

The principal advantages to describe an object by a set of feature vectors instead of by only one are that: 1) the information is distributed, 2) several feature vectors describe a pattern, 3) patterns from different classes share common features and 4) variations among the different elements of one feature vector will be the same.

Now, given an initial feature vector, the problem is how to divide it into several SFVs without supervision. Until now this is done by hand.

A feature vector can be split into different SFVs contributing to the description of the object, f.e. to describe a polygon we can use a SFV to describe the types of vertices, another SFV to describe the angles between vertices, and so on.

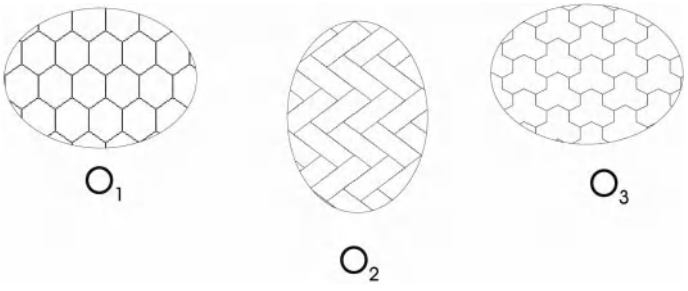


Fig. 1. A feature vector is used to describe each object.

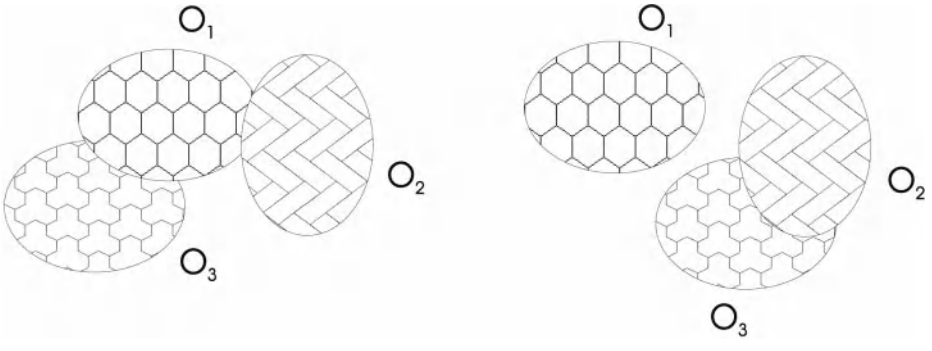


Fig. 2. A set of features vectors is used to describe a set of objects.

Another way is to split the feature vector into several SFVs grouping together attributes with similar values.

Suppose thus that an object is described in terms of several SFVs. The problem now is how we can use this information to get a classification using supervised or unsupervised classifiers if the patterns share common features? At this point, we have the sets of SFVs for the different objects, but we do not know how many types of features we have, i.e., how many clusters for each SFV exist. f.e. in a face recognition problem we have the SFVs describing the eyes, but we do not know how many types of eyes we have. We would like to have a procedure to automatically find these clusters.

In this paper we propose to use the ART2 to obtain the clusters of the different SFVs describing the objects. For each SFV we need an ART2 and a vigilance threshold ρ.

In the following example we show how we can use the ART2 to generate clusters of features (prototypes). Seven different objects are shown on Figure 3 with different textures. If the objects are described by the separation between lines and the thickness of the lines, and for a given threshold we may get the results shown on Table 1, but if we select a greater threshold we then may get the results shown on Table 2.

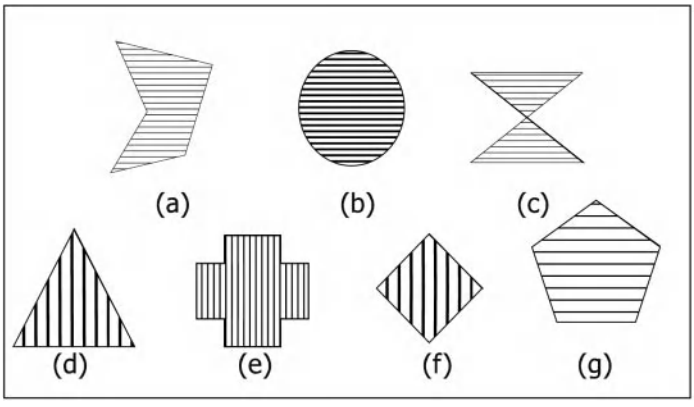


Fig. 3. Different objects with different thickness and separation between lines.

Table 1. *Clusters obtained when the vigilance threshold is small.*

clusters	objects
1	a,c
2	b,e
3	d,f
4	g

Table 2. *Clusters obtained when the vigilance threshold is greater.*

clusters	objects
1	a,b,c,e
2	d,f,g

From this example, we can see that the clusters obtained depend on the threshold value, but there is no way to select the right threshold value. In the next section we present a first way to obtain these values. No method had been given until now for this purpose.

3 Threshold Selection

So far the ART2 NN has been used for unsupervised learning. In this paper we also propose how to use it for supervised learning, and how to select a suitable threshold value.

Just to remember, the functioning of the ART2 NN is based on the selection of the vigilance threshold ρ. If a small value for ρ is selected, then similar features will be put in different clusters and so a lot of clusters will be generated. In the other hand, if a greater value for ρ is selected, then many clusters will be obtained; in this case different features will be merged into the same cluster.

If an object is described by a feature vector then just a threshold value should be selected. In the contrary, if the object is described by several SFVs, then a threshold value for each SFV should be selected.

To select a suitable threshold, the following procedure is proposed with the aim to increase the intraclass distance and to decrease the interclass distance.

If an object O_i belongs to class C_n, and an object O_j belongs to a different class C_m, then object O_i is differentiated from object O_j if they are classified into different classes, i.e.

$$O_i \in C_n \qquad O_j \in C_m \qquad \text{for } n \neq m \tag{2}$$

They get confused if they are clustered together.

$$O_i \in C_n \qquad O_j \in C_m \qquad \text{for } n = m \tag{3}$$

Now, if CD is the number of objects from different classes that can be differentiated, and CC the number of objects from the same class that are missclassified. The procedure to get the threshold value for supervised learning is the following.

Procedure: Selecting the thresholds for supervised learning
Input: N sets of SFVs
Output: N thresholds, one for each SFV
{
For each SFV i

Step 0. Initialize $CD = 0$, $CC = 0$;

Step 1. Calculate the similarity (ED) between the SFVs belonging to different classes, and the similarity (EC) between the SFVs belonging to the same class as follows:

$$ED_j = |u - v| \quad u \in C_n \quad v \in C_m \qquad n \neq m, j = 1, ..., M$$
$$EC_j = |u - v| \quad u, v \in C_n \qquad j = 1, ..., N$$

Step 2. Select an initial threshold value $\rho_i(0)$ as follows:

$$\rho_i(0) = \frac{1}{M} \sum_{j=0}^{M} ED_j$$

Step 3. Two objects belonging to different classes are differentiated with ρ_i if:

$$(ED_j > \rho_i(t)) \qquad j = 1, ..., M$$

if this is true, then increase CD by 1.

Step 4. Two objects that belong to the same class get confused (missclassified) with ρ_i if:

$$(EC_j > \rho_i(t)) \qquad j = 1, ..., N$$

if this is true, then increase CC by 1.

Step 5. $\rho_i(t+1) = \rho_i(t) + \Delta$
Step 6. Initialize $CD = 0$, $CC = 0$ and goto step 3
Step 7. Select the value of ρ_i for that SFV i such that:

$$\max_k |CD_k - CC_k|$$

}

This procedure selects the value that gives the greatest differentiation between objects belonging to different classes, giving at the same time the least confusion (missclassification) between objects belonging to the same class. The aim is to maximize CD and to minimize CC.

Let $v = CD - CC$,

If $v = 0$ then the threshold value does not differentiate, confusing thus among classes.

If $v > 0$ then the threshold value discriminates better, giving a poor confusion among classes.

If $v < 0$ then the threshold value confuses more, giving a poor discrimination.

These 3 cases indicate that a positive value exhibits a better discrimination, while a negative value shows a greater confusion; a value near to zero does not discriminate, neither it confuses. Considering these cases, we conclude that the criterion to be maximized should be positive and as greater as possible.

Unsupervised learning uses feature vectors without class labels, so the proposed criterion for supervised learning can not be used. For unsupervised classification the following criterion should be used:

$$\max_k \ CD_k \tag{4}$$

The value for CC is not used here because in unsupervised learning the class labels of the objects are unknown and so CC can not be computed.

The procedure for unsupervised learning considers that all the objects belong to different classes, so we do not have objects belonging to the same class. The procedure to get the threshold value for unsupervised learning is the following.

Procedure: Selecting the thresholds for unsupervised learning
Input: N sets of SFVs
Output: N thresholds, one for each SFV
{

For each SFV i

Step 0. Initilaize $CD = 0$

Step 1. Calculate the similarity (ED) between the different objects on the training set:

$$ED_j = |u - v| \quad u \in C_n \quad v \in C_m \qquad j = 1, ..., M$$

Step 2. Select an initial threshold value $\rho_i(0)$ as follows:

$$\rho_i(0) = \frac{1}{M} \sum_{j=0}^{M} ED_j$$

Step 3. Two objects differentiate given a ρ_i if:

$$(ED_j > \rho_i(t)) \qquad j = 1, ..., M$$

if this is true, then increase CD by 1.

Step 4. $\rho_i(t+1) = \rho_i(t) + \Delta$

Step 5. Initialize $CD = 0$, and goto step 3

Step 6. Select the value of ρ_i for that SFV i so that:

$$\max_k \ CD_k$$

}

At the end of this procedure a set of threshold values, one for each SFV will be obtained. These threshold values can be further used to train each ART2 NN.

In the following section we provide an example of how to use the proposed procedures to select the threshold values. Results in the case of objects without noise and objects with noise are presented.

4 Example of Application: Character Recognition

In this very simple example, 3 similar characters were selected to show the performance of one of the procedures described in the last section. In this case the training set is composed of only one object for each class (Figure 4), so the problem is handled as an unsupervised classification problem because a set of objects belonging to the same class is not given, i.e., $CC = 0$.

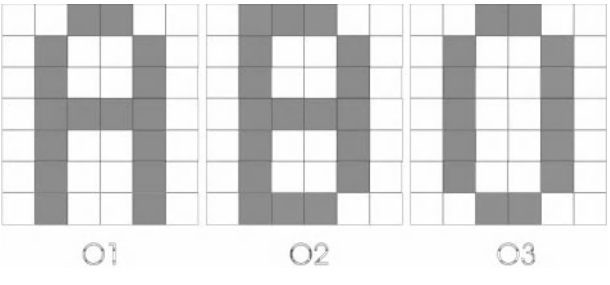


Fig. 4. The training set.

In the remaining sections, the different SFVs will be proposed and extracted. The threshold values for each SFV are then obtained. Then for each SFV, the ART2 NNs are trained. Finally, the performance of the trained ART2 NN is tested in the case of noisy objects.

4.1 SFVs Extraction

Before extracting the feature vectors, we need to select the attributes that will describe the different objects that are going to be classified. In this case each character is described as follows. For each row the number of black cells is computed, as shown on Figure 5, the same process is repeated for each column [6].

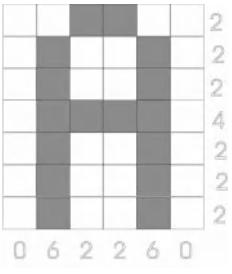


Fig. 5. Feature vector extraction for character "A".

The normalized features vectors for the objects shown on Figure 4, are the following:

O1	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{4}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{0}{7}$	$\frac{6}{7}$	$\frac{2}{7}$	$\frac{2}{7}$	$\frac{6}{7}$	$\frac{0}{7}$
O2	$\frac{3}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{3}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{3}{6}$	$\frac{0}{7}$	$\frac{7}{7}$	$\frac{3}{7}$	$\frac{3}{7}$	$\frac{4}{7}$	$\frac{0}{7}$
O3	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{2}{6}$	$\frac{0}{7}$	$\frac{5}{7}$	$\frac{2}{7}$	$\frac{2}{7}$	$\frac{5}{7}$	$\frac{0}{7}$

As discussed at the beginning of section 3, if the ART2 NN is used with these features vectors, then its performance will depend on the selected vigilance threshold. In the worst case, and depending on the threshold value, the ART2 NN will generate one cluster, in the best case, it will generate 3 clusters. If we use only one class, thus two of the three objects will be incorrectly classified. To avoid this, we propose to use a set of SFVs instead of only one feature vector. In this case we suggest to use two SFVs. One SFV for the horizontal features (v_A), and one for the vertical features (v_B). The two SFVs are the following:

$$v_{A1} = \left[\frac{2}{6}, \frac{2}{6}, \frac{2}{6}, \frac{4}{6}, \frac{2}{6}, \frac{2}{6}, \frac{2}{6}\right] \qquad v_{B1} = \left[\frac{0}{7}, \frac{6}{7}, \frac{2}{7}, \frac{2}{7}, \frac{6}{7}, \frac{0}{7}\right]$$
$$v_{A2} = \left[\frac{3}{6}, \frac{2}{6}, \frac{2}{6}, \frac{3}{6}, \frac{2}{6}, \frac{2}{6}, \frac{3}{6}\right] \qquad v_{B2} = \left[\frac{0}{7}, \frac{7}{7}, \frac{3}{7}, \frac{3}{7}, \frac{4}{7}, \frac{0}{7}\right]$$
$$v_{A3} = \left[\frac{2}{6}, \frac{2}{6}, \frac{2}{6}, \frac{2}{6}, \frac{2}{6}, \frac{2}{6}, \frac{2}{6}\right] \qquad v_{B3} = \left[\frac{0}{7}, \frac{5}{7}, \frac{2}{7}, \frac{2}{7}, \frac{5}{7}, \frac{0}{7}\right]$$

These SFVs conform the training set; for each SFV a threshold value should be obtained. Due to each class contains only one object, the procedure for unsupervised classification will be used. Remember that in unsupervised classification the training set considers that each object belongs to a different class.

4.2 Threshold Selection

As we mentioned the procedure used to select the threshold value depends on the problem. For the three character recognition problem we are trying to solve, we use the procedure for unsupervised classification.

According to the procedure, the value for $\rho_i(0)$ is computed for each SFV. These values are shown in bold on Table 3. The range for ρ_1 and ρ_2 considered is between $(0.5 * \rho_i(0))$ to $(1.5 * \rho_i(0))$, this means that $\Delta = (0.1 * \rho_i)$, as shown on Table 3.

Table 3. *Results for the objects on Figure 4.1.*

ρ_1	**CD**	**Clusters**	ρ_2	**CD**	**Clusters**
0.223	3	3	0.286	3	2
0.268	3	3	0.343	3	2
0.313	3	1	0.401	2	1
0.357	2	1	0.458	2	1
0.402	2	1	0.515	2	1
0.447	**2**	1	**0.573**	**2**	1
0.491	2	1	0.630	2	1
0.536	0	1	0.687	2	1
0.581	0	1	0.744	0	1
0.625	0	1	0.802	0	1
0.670	0	1	0.859	0	1

For each value of ρ_i the corresponding values of CD are also computed (Table 3). Column 3 and 6 show the number of clusters obtained for each value of ρ_1 and ρ_2.

We can see from Table 3, that the values maximizing $\max_i CD_i$ are: $\rho_1 = 0.223$ and $\rho_2 = 0.286$. In this case the first maximum value was chosen as the value for ρ_i. With this values, then the ART2 NNs can be trained. The clusters obtained using both ART2 NNs are shown on Tables 4 and 5, respectively. As we can see for SFV v_A, one cluster for each training object was created; for v_B just two clusters were obtained. This means that object 1 and object 3 share one SFV v_B, and so they are clustered together.

As we know, in both supervised or unsupervised classification the descriptions of an object given by a set of SFVs are given. In both cases we know to which object each SFV belongs as we can see on Tables 4 and 5. This is done by means

Table 4. *Clusters obtained for the SFV v_A using the ART2 NN with $\rho_1 = 0.223$.*

Clusters for v_A	**Objects**
C_{a1}	O_1
C_{a2}	O_2
C_{a3}	O_3

Table 5. *Clusters obtained for the SFV v_B using the ART2 NN with $\rho_1 = 0.286$.*

Clusters for v_B	**Objects**
C_{b1}	O_1, O_3
C_{b2}	O_2

of the ART2 NN. This characteristic of the ART2 NN is used in this paper to get a list of candidates sharing common features, in both supervised or unsupervised classification.

4.3 Results

Once the threshold values for ρ_1 and ρ_2 have been obtained, and the ART2 NNs have been trained, we can proceed to test their performance to classify new objects.

If we consider the original objects shown on Figure 4, the classification results are shown on Table 6. Column 2 shows the clusters selected by SFV v_A, column 3 shows the clusters selected by SFV v_B. Column 4 shows the objects belonging to that cluster, meaning that f.e. for pattern P1, object O1 was invoked by C_{a1} and C_{b1} and object O3 was invoked by C_{b1}. Finally column 5 shows the objects that were invoked the most during the classification process (marked in bold). Together with the desired objects other objects were also selected.

From this example we can see one of the advantages of using a set of SFVs instead of only one. With several SFVs we get a list of candidates sharing common features, so the output is given by a list of candidates sharing the same feature. Finally the output will give the objects that were invoked the most, given in a list of final candidates.

Table 6. *Results obtained for the objects shown in Figure 4.1.*

	ρ_A= **0.223**	ρ_B= **0.286**	**Output**	**Final Output**
P1	C_{a1}	C_{b1}	O1, O1, O3	**O1** o O3
P2	C_{a2}	C_{b2}	O2, O2	**O2**
P3	C_{a3}	C_{b1}	O3, O1, O3	O1 o **O3**

Now consider the case of noisy objects, as those shown on Figure 6. The results are shown on Table 7. Again, as we can see from this table that using several SFVs instead of only one, will give us the opportunity to have different classifications from different sources. So instead of getting a unique output, the output will be a list of candidates, selected by the different clusters. Each cluster selects the objects sharing common features. For example, on Table 7, for pattern P3, cluster C_{a3} and cluster C_{b1} were selected, so as we can see on Tables 4 and 5, C_{a3} invokes object O3, and C_{b1} invokes objects O1 and O3. Finally O3 was invoked two times and O1 was invoked just once. The final list of candidates

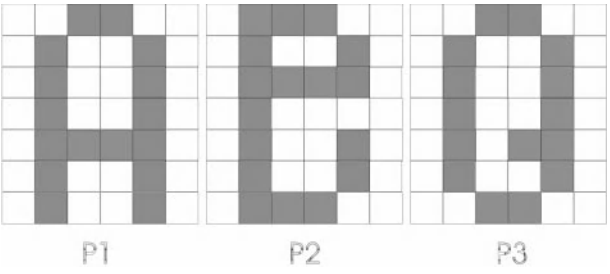


Fig. 6. Objects with noise.

Table 7. *Results obtained for the objects shown on Figure 4.3*

	ρ_A= **0.223**	ρ_B= **0.286**	**Output**	**Final Output**
P1	C_{a3}	C_{b1}	O3, O1, O3	O1 o **O3**
P2	C_{a2}	C_{b2}	O2, O2	**O2**
P3	C_{a3}	C_{b1}	O3, O1, O3	O1 o **O3**

shown on column 5, shows that the desired output O3 is the object with the highest probability to correspond to the input pattern.

If only one SFV was considered then the output for P3 would be O1 or O3 but not both at the same time, giving as a result a missclassification.

Another example is shown on Figure 7, in this case the objects with highest probability correspond to the desire patterns (Table 8).

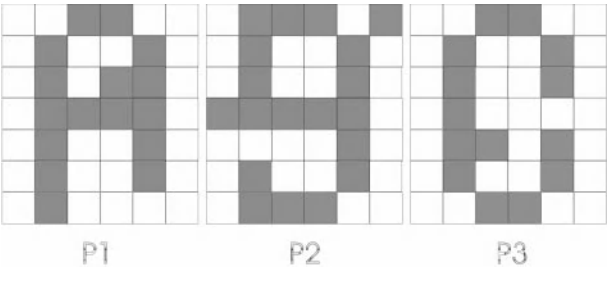


Fig. 7. Objects with noise.

Table 8. *Results obtained for the objects shown on Figure 4.4*

	ρ_A= **0.223**	ρ_B= **0.286**	**Output**	**Final Output**
P1	C_{a1}	C_{b1}	O1, O1, O3	**O1** o O3
P2	C_{a2}	C_{b2}	O2, O2	**O2**
P3	C_{a3}	C_{b1}	O3, O1, O3	O1 o **O3**

5 Conclusion and Future Work

In this paper we proposed a first procedure to automatically select the vigilance threshold for the ART2 NN. We also show how this procedure can be used for both supervised and unsupervised classification.

Also, a set of SFVs to describe a pattern, instead of only one feature vector was proposed. The advantages to use several SFVs are that the information is distributed, and a better performance is obtained, and that instead of having a candidate at the output, we get a list of ordered candidates in terms of the times they were invoked by the SFVs. This decreases missclassification.

The effectiveness of the procedure was tested in the scenario of character recognition. Three simple characters were only used. The results were satisfactory although the objects were too similar. This could not had been possible if the objects were described by only one feature vector.

Until now the set of SFVs is selected by hand. Actually we are working toward the development of an automatic technique to also select a set of SFVs given only one feature vector.

We pretend to use the proposed procedure in several unsupervised and supervised classification problems, f.e. face recognition.

Acknowledgments. This research was supported by the Centro de Investigación en Computación of the IPN.

References

1. G.A. Carpenter and S. Grossberg (1987). ART2: self-organizing of stable category recognition codes for analog input patterns. Applied Optics, 26, pp. 4919-4930.
2. R. Schalkoff (1992). Pattern Recognition: statistical, structural and neural approaches. John Wiley & Sons, Inc.
3. J. A. Freeman and D. M. Skapura (1992). Neural Networks: algorithms, applications and programming techniques. Addison Wesley.
4. A. S. Pandya and R. B. Macy (1995). Pattern Recognition with Neural Networks in C++. IEEE Press.
5. G. Francis (1996). The stability-plasticity dilemma in competitive learning. Department of Psychology Sciences. Purdue University. Technical Report No. 96-1.
6. C. G. Looney (1997). Pattern Recognition using Neural Networks. Oxford Universtiy Press.

Investigation of the One-Armed Bandit by Neural Network

Frank C. Lin and Xiaojiang Ju

Dept. of Math and Computer Science
University of Maryland Eastern Shore
Princess Anne, MD 21853-1299

Abstract. We conducted two experiments with the One-Armed Bandit, one at the VFW lodge at Salisbury, MD. and another at the Taj Mahal Casino and Hotel, at Atlantic City. We trained a backpropagation Neural Network using the patterns of hundreds of successive pulls as input. We show that is is possible to predict, by a trained Network, the pattern of the ensuing combination.

1 Introduction

The One-Armed Bandit, also known as the slot machine, was invented by a German immigrant Charles Fey in San Francisco in 1899. the original symbols used were playing card suits. At the turn of the century the manufacturer, Herbert Mills, introduced symbols for fruits such as cherries, plums and lemons, as well as the bar and the bell. He also added the jackpot. [Patricks, 1994] [Halcombe, 1996].

If a slot machine has three reels with twenty symbols on each reel, your chance of winning the jackpot is 8,000 to 1. It cannot be overemphasized that the player has no control whatsoever over the destiny of the reels. The player's only input is to pull the arm, or with electronic machines, push a button. In the 1940's an Idaho farmer discovered that all the reels on a fruit machine made exactly the same number of turns. thus he was able to manipulate the reels to win consistently. However, in 1951 a device called the variator was added to the slot machine to randomize the outcome. Since then no one has succeeded to decode the sequence.

However, it is clear that no slot machine can be purely random, since the casino operator programs a bias into the machine to insure a profit. In Britain, the minimum payback from a machine on premises licensed by the Gaming Board is 75%. in the United States, the Johnson Act passed by Congress prohibits slot machines in all states except Nevada and New Jersey (and some special locations). In New Jersey, the minimum payback percentage is 83 whereas in Nevada it is 75%. Some non-profit organizations, such as the Veteran of Foreign Wars, are exempt. The payback percentage of slot machines on the floor may vary, although a minimum overall payback is mandated.

The aforementioned bias is implemented by a computer chip, which may be installed at a central server connected by telephone lines. The microprocessor is driven continuously irrespective of whether the One-Armed Bandit is in use. Its output sends signals every fraction of a second and directs the Bandit to stop at the appropriate symbol. New machines may have more than 20 symbols in a reel.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 401—404, 2000.

Without the foresight obtained by the Neural Network, the key to beating the One-Armed Bandit is to quit as soon as you win. Psychologically, however, the player is enticed to continue gambling in order to win more Indeed, to beat the machine psychologically, one must embark upon the game with the determination to lose. (Other strategies are possible).
An One-Armed Bandit without the arm is shown in Fig. 1:

Fig. 1 The One-Armed Bandit

The three reels are activated when a coin is deposited and the arm is pulled or a button pushed. These three reels spin for several predetermined seconds, and come to rest, not simultaneously, but one after the other. Nine symbols are visible at the window, and depending on the machine, the middle line determines the payoff. Other than the jackpot, there are smaller payoffs such as two cherries, three plums, etc.

In general it is reasonable to assume that the time evolution of the One-Armed Bandit in its phase space of dimension n, which takes into account the number of reels and the number of symbols, follows a nonlinear differential equation of motion. In practice, due to intrinsic constraints, symmetries, etc. The system develops in a non Euclidean asymptotic submanifold of much lower dimension called an attractor. Neural Networks, however, can only perform mappings between Euclidean spaces. Nonetheless, it has been shown by Takens [Takens, 1981] and others that the phase space information exhibited by our synamic system can be reconstructed from a single time series and its derivatives by embedding the submanifold in an Euclidean space of dimension m, shere $n >= 2m + 1$. In practice, the simulation of this time series by a Neural Network, perhaps due to its nonlinear character, has been extraordinarily successful (See e.g. [Elborn and Lin, 1992], [Lin and Lin, 1993], [Ansari and Lin, 1994], [Lin and panikkar, 1995], [Johnson and Lin, 1995], [Baptist, Lin and Nelson, 1996]. In most cases its performance is superior to conventional statistical methods such as Box Jenkins.

2 Experiment # 1:

This experiment, of an exploratory nature, was conducted at the VFW lodge at Salisbury, MD. The recording of the experimental data was accomplished manually. Casinos everywhere are skeptical if the customer is armed with a laptop, and ensuing

exchanges could prove detrimental with regard to the success of the project. The symbols were coded with numeric constants between zero and one. For instance, the lemon is assigned the value 0.0, the cherry the value 0.6, etc. (See Table I). By experiment, the optimal configuration of the backprop Neural network was found to be 9-37-17-3, where the input node represent the nine symbols that are visible through the window, and the output node represent the winning (or losing) combination in the middle horizontal line. Altogether 63 training patterns were used to calculate the sum of differences between the output and the target to check the goodness of fit, following which a projection to the next combination was made. It was ascertained that the prediction was, within tolerance limits, indeed correct.

Table 1. : Numeric Representation of Symbols. (VFW lodge)

Code	Symbol	Code	Symbol
.0	Lemon	.1	Triple Bar
.2	Single Bar	.3	Double Bar
.4	Watermelon	.5	Grape
.6	Cherry	.7	Seven
.8	Plum	.9	Bell

3 Experiment # 2:

This experiment was conducted at the Taj Mahal Casino and Hotel, Atlantic City. Our design is similar to the above, but we greatly enhanced the experimental data to 1100 patterns, which called for over six hours of continuous operation of the slot machine. We have reason to believe that this may be an overkill. In the process a total of approximately $15 was lost in the interest of scientific inquiry. The code for this machine is listed in Table II.

Table 2. Numerical Representation of Symbols. (Taj Mahal)

Code	Symbol	Code	Symbol
.1	Blue Seven	.2	Red Seven
.3	Red Cherry	.4	Blue Bar
.5	Orange Bar	.6	Red Bar
.7	Nickel Sign	.8	Blank Space

In this experiment we included not just the middle horizontal line but all nine symbols that are visible through the window in the output. We also investigated the possible imopact of network configuration by varying the number of hidden units. Thus we implemented the topologies 9-X-9, where X = 2, 3, 4 and 6.

Referring to Table III, it is seen that the prediction with X = 2 was incorrect. For X = 3, the prediction was 100% correct for all nine symbols. For X = 4 and 6, the prediction was accurate about 90% of the time, and the middle payback line was always exactly duplicated. It is also noteworthy that the convergence rate needs not to be very high for the Neural Network to capture the pattern generated by the slot machine.

The values adopted for bias, the learning rate and the momentum coefficient were 0,0.2, and 0.4 respectively

4 Conclusion:

Without any detailed knowledge of the internal mechanism of the One-Armed Bandit, We investigated the posibility that the next combination can be predicted using a Neural Network based on past performance. The answer is affirmative in the two machines we tested. It is conceivable that this insight can be exploited for pecuniary gain. Other casino games may also be scrutinized using this paradigm.

Table 3.: Result of the Taj Mahal Production Run

Nodes in Hidden Layer	Tolerance	Conver-gence-rate(%)	Predicted output	Real output	Predict-rate(%)
X=2	0.04	895/1095 =81.7%	.500 .600 .400 .400 .400 .400 .500 .700 .100	.300 .600 .000 .000 .000 .600 .600 .700 .000	2/9 =22.2%
X=3	0.04	85/1095 =7.76%	.300 .599 .039 .024 .029 .600 .600 .701 .025	.300 .600 .000 .000 .000 .600 .600 .700 .000	9/9 = 100%
X=4	0.04	481/1095 =43.9%	.300 .999 .005 .003 .003 .600 .600 .700 .0002	.300 .600 .000 .000 .000 .600 .600 .700 .000	8/9 =88.9%
X=6	0.047	485/1095 =44.3%	.300 .999 .005 .004 .004 .600 .600 .700 .005	.300 .600 .000 .000 .000 .600 .600 .700 .000	8/9 =88.9%

Acknowledgement: We thank Gustavo Limon for contributions to Experiment # 1.

References:

[Ansari and Lin, 1994] Aslam M. Ansari and Frank C. Lin. Forecasting Financial Trends in Stock Markets with Neural Networks. Proceedings of the 3rd International conference on System Integration, pages 188-196, Sao Paulo, August 15-19 1994.

[Baptist, Lin and Nelson, 1996] Garry Baptist, Frank C. Lin and james Nelson Jr. Note on the Long Term Trend of the Dow Jones Industrial Average. Neural Network World, 6(3): pages 259-262, March 1996. IEEE European Workshop Special Issue on Neurofuzzy '96.

[Elborn and Lin, 1997] Shelby Elborn and Frank C. Lin. Analysis and Prediction of Earthquakes for Southern California Using a Neural Network. Geologia Applicata eldrogeologia. Vol.xxvii, pages, 111-121, Bari, 1992.

[Halcombe, 1996] Slot Smarts. Carol Publishing Group, Secaucus, N.J., 1996.

[Johnson and Lin, 1995] Gregory P. Johnson and Frank C. Lin. Hurricane Tracking via Backpropagation Neural Network. In Proceeding of the IEEE International Conference on Neural Networks, Vol.2, pages 1103-1106. Perth, Australia. 1995.

[Lin and Lin. 1993] Frank C. Lin and Mei Lin. Analysis of Financial Data Using Neural Network. *A.I. Expert*, Feb. page 37-41. San Francisco. 1993.

[Lin and Panikkar. 1995] Frank C. Lin and Indira Panikkar. Study of Seismic Activity in Central Asia Applying a Parallel distributed Paradigm. *Proceedings of the 1995 International Geoscience and Remote Sensing Symposium*. July 10-14. Firenze, Italy, 1995.

[Patricks, 1994] John Patricks. *Slots*. Carol Publishing Group, Secaucus, N.J. 1994.

[Takens. 1981] F. Takens. Detecting Strange Attractors in Turbulence. *In Lecture Notes in Mathematics*, Vol.898 p366 (Springer, Berlin), D.A.Rand and L.S.Young, Eds.1981.

Knowledge Base System for Diagnostic Assessment of Doppler Spectrogram

B. Das[1], S.K. Mitra[2], and S. Banerjee[1]

[1] Dept of Electronics & ECE, Indian Institute of Technology
Kharagpur, West Bengal, India

[2] EKO Heart Foundation
Calcutta, West Bengal, India

Abstract. The aim of this paper is to develop an automated diagnosis system for arterial diseases from the Doppler ultrasonography spectrogram. A feature-based classification of Doppler spectrogram has been done for its clinical assessment. Development of the automated diagnostic tool for Doppler spectrogram involves three steps (a) feature extraction, (b) classification of spectrogram based on clinical symptoms and (c) diagnosis of arterial condition of the concerned region. Artificial Neural Network is used for classification of the spectrograms. Arterial condition of a specified region is evaluated from symptoms obtained from a number of spectrograms in the different nearby regions. Bayesian probabilistic method is used for diagnostic evaluation of the arterial status from the obtained spectrogram. The results satisfied 83% of the obtained cases.

Keywords: Doppler ultrasonography, spectrogram, Bayesian statistics, Neural Network, Ischemia, Vasodilatation, Stenoses.

1 Introduction

With advent of computer technology, image processing and automated classification techniques have found application in analysis of medical signals. Many techniques have been followed for the classification of biomedical images. Similarity searching in medical images is a widely used technique for image analysis and classification [1]. Automated classification using knowledge base technique has been studied in a number of biomedical images. Neural network is successfully implemented as pattern and statistical classifiers [2]. It has been used for the detection of breast cancer, real-time ischemic episodes or medical diagnosis of CT liver scan image or evaluation of parenchymal echo pattern of cirrhotic pattern and chronic hepatitis [3].

Study of Doppler spectrogram unveils many facts about the arterial nature of blood flow and cardiac status. The Doppler spectrum was displayed by Bommer et al. [4]. Diagnostic assessment of the arterial system using Doppler spectrum is generally based on the interpretation of the measured spectrum and its cyclic variation. To simplify the process, various indices are often used to quantify the individual spectra, and these can yield information that correlates closely to the disease present. Feature based diagnosis of the Doppler spectrum was reported in 80's [5]-[9]. The study of

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 405—416, 2000.

arterial diseases from the Doppler spectrum was done with the calculation of transfer function in time domain [9]. However, the simultaneous measurement of the input and output blood velocity versus time waveform is incapable of correctly describing changes in normal physiology of limb circulation. More powerful method was developed for describing shape of the blood-velocity versus time waveform in terms of Laplace transform [10]-[12]. The extent of arterial disease is determined by recognizing patterns in the spectral and temporal characteristics of the backscattered Doppler signal. Feature based recognition and classification of Doppler spectrograms use various quantified features to identify the spectrogram.

Clinical assessment of the Doppler spectrogram is presented in this paper using artificial neural network. Final diagnostic assessment is made from the symptoms obtained from a number of spectrograms of different nearby regions using a probabilistic approach.

In the next section the paper describes the features used to describe the spectrogram and the effect of the arterial factors on blood velocity and eventually on the features. Feature based classification technique using artificial neural network is shown in section 2. In section 3 the algorithm for end diagnosis. Section 4 shows the results leading to the diagnosis from a set of correlated spectrogram. Section 5 finally concludes the paper.

2 Clinical Assessment of Doppler Spectrogram

The feature-based diagnosis developed from the Doppler spectrogram involves (i) extraction of features (indices) from the obtained spectrograms, (ii) Classifying the spectrogram in the proper cluster and (iii) Diagnosing the ailment from a set of correlated spectrograms.

2.1 Feature Extraction

Several characteristics of spectral waveform are analyzed to interpret and classify them [13]-[15]. The peak values are important as they give the velocity information. The change of velocity with time is also featured in the shape of the waveform. Forward flow is normally featured on the top half of the baseline and retrograde flow on the bottom half. The characteristics, which feature the spectrogram, are both the boundary as well as the regional features. The boundary features include acceleration slope, deceleration slope, presence of negative peak, systolic to diastolic ratio, period. Spectral broadening index, and the moment features like the coefficient of kurtosis, coefficient of skewedness constitute the regional features. The effect of blood flow on the features may be seen as in [15]-[17], and are discussed in the subsequent part.

Proximal to the stenosis a normal or lightly blunted waveform may be seen. The peak frequency should be normal, and there may or may not be a rounding of the peak. The separation between the first and second components may disappear because of vascular elastic changes along the stenosed area. The window may show some flow scattering throughout

Across the stenosis an increased-frequency (higher velocity) waveform will usually be seen. The same characteristics as in the proximal waveform may be noted, and the envelope may begin to break up. The velocity usually increases directly with the degree of stenosis. The window may also begin to disappear as spectral broadening increases. A corresponding increase in pitch is heard with the Doppler. Flow remains nearly constant until a hemodynamically significant stenosis (at 60% to 70% luminal reduction) occurs.

Immediately distal to stenotic area, disturbed or turbulent flow may be found. Peak systolic velocity and frequency increase as the flow stream increases by the jet effect, originating at the stenosis. An area of flow stagnation that occurs immediately distal to the plaque or lesion near the artery wall may cause turbulence during systole. The spectral waveform will have a ragged-appearing indistinct envelope, with widely distributed frequencies, and a completely absent window. Turbulence and eddy formation cause an increase in higher-energy intensities at the baseline and in severe cases may cause a completely disrupted waveform with apparent velocity reductions at systole in bidirectional analysers.

A signal obtained more distally from the stenosis, away from the turbulent region, may assume the appearance of a reduced waveform with even intensity and broadening throughout, disturbed envelope edges, and no visible flow components. The peak frequency and velocity may be markedly diminished compared with a peak of the waveform seen directly at the level of stenosis. The peak frequency may be markedly rounded. Severely diminished flow may almost become flat in appearance.

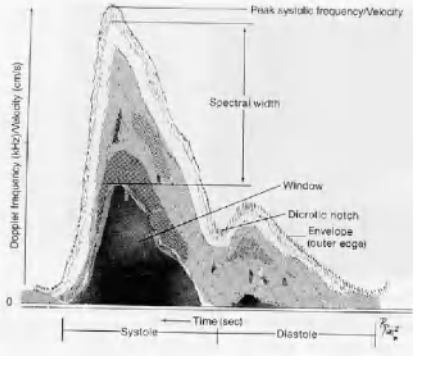


Fig 1. Regions and sections in a spectrogram

In total occlusion a flow may actually drop to or below the baseline, and a very irregular-appearing spectral waveform with no distinguishing characteristics may be seen. This finding of high-resistance flow is also suggestive of a distal obstruction especially when seen in a patent segment of a normally low resistance vessel, such as the internal carotid artery.

The characteristics of flow increase across a stenosis, are also seen for extremity arteries as well. Distal to the stenosis a rounded blunt waveform will be noted, with disappearance of the window and evidence of low flow velocities. As flow decreases a flattened signal close to the baseline will usually be found.

In addition to the aforesaid parameters, changes in the instantaneous Doppler spectrum may be helpful in detecting stenoses. As has been mentioned previously, at

$$SBI = \frac{f_{max} - f_{mean}}{f_{max}} = 1 - \frac{f_{mean}}{f_{max}} \quad (1)$$

a point near to stenosis there may be a disturbed flow which results in broadening of Doppler spectrum. The relative changes in the Doppler spectrum for each spectral line were quantified by the calculation of spectral broadening index (SBI) [13], where

The SBI is calculated at peak systole unless otherwise indicated.

Apart from these parameters there are other feature parameters like Mean velocity, Pulsatility index, Resistive index, and the moment features which are given by :

1. MV (Mean Velocity) = (A + 2B)/3 ;
   where A and B are the peak systolic and end diastolic values respectively.
2. PI (Pulsatility Index) = A-B/MV ;
   Signal recorded from a vessel may demonstrate a damped, less pulsatile waveform providing evidence of proximal stenosis – not obvious from velocity value alone. PI is a common measure for describing the shape of signal waveform.
3. RI (Pourcelet's resistive index) = (A-B)/A ; It is a measure of the peripheral flow resistance. High diastolic flow velocities resulting in a low RI value characterize low vascular resistance.
4. As is the case of any waveform the moment features give a measure of the flow disturbance nature [14].

- CV (Coefficient of Variance) =

$$\frac{\sigma}{|\mu|} \quad (2)$$

It is a normalized measure of spread or spectral width about the mean. It is calculated by dividing the standard deviation (σ) by the absolute mean frequency (μ).

The parameters like σ and μ are defined as :

$$\mu = \frac{\sum_{n=0}^{N} A_n f_n}{\sum_{n=0}^{N} A_n} \quad (3)$$

$$\sigma^2 = \frac{\sum_{n=0}^{N} (f_n - \mu)^2 A_n}{\sum_{n=0}^{N} A_n} \quad (4)$$

where A_n is the amplitude corresponding the frequency f_n.

For non-disturbed flow CV is small since there is a narrow spread of frequencies at the higher end.

- CS (Coefficient of Skewedness) =

$$\frac{m3}{\sigma^3} \tag{5}$$

measures the amount of skewedness or asymmetry about the mean in a random distribution. It is derived from the third moment `m3' about the mean divided by σ^3. For non-disturbed flow CS is negative.

$$m3 = \frac{\sum_{n=0}^{N} (f_n - \mu)^3 A_n}{\sum_{n=0}^{N} A_n} \tag{6}$$

- CK (Coefficient of Kurtosis) =

$$\frac{m4}{\sigma^4} - 3 \tag{7}$$

where 'm4' is the measure of the fourth moment and CK describes the peakedness of the distribution. It is calculated by dividing the fourth moment about the mean by σ^4 and subtracting 3.

$$m4 = \frac{\sum_{n=0}^{N} (f_n - \mu)^4 A_n}{\sum_{n=0}^{N} A_n} \tag{8}$$

These features add a degree of precision to the diagnosis.

The analysis is carried over a region of interest (ROI) and all the feature values are taken as average of the features extracted over the ROI.

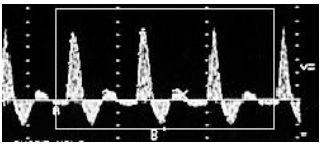


Fig 2. Region of Interest selected from a spectrogram

2.2 Classification of Spectrograms

The spectrograms are classified using an artificial neural network (ANN). Backpropagation neural network (BPNN) has been successfully adopted for various biomedical image classification. Here feature based classification is done for the spectrograms. The extracted features discussed in the last section are set as input to the BPNN network. Spectrograms are classified into five classes corresponding to (i) Normal, (ii) Distal, (iii) Ischemic, (iv) Peripheral Vasodilatation, and (v) Vasoconstriction.

The hyperspace for classification is not of linear pattern, rather of an unknown non-linear pattern. The transfer function that is used here is the sigmoid or S-shaped curve. The curve approaches minimum values or maximum values at the asymptotes. Mathematically, the interesting feature of this curves is that both the function and its

$$y = \frac{1}{1 + \exp(-x)} \tag{9}$$

derivatives are continuous. This option works fairly well for non-linear partitioning and convergence of the series. Sigmoid function is mathematically represented as :

The rate at which the ANNs learn depends on several controllability factors. A slower rate means a lot of time is spent in accomplishing off-line learning to produce an adequate trained system. With faster learning rate the network may not be able to make fine discriminations possible. Thus an optimum value of learning rate is chosen for training the ANN. Here we have adopted an adaptive learning rate η which modifies with the mean square error. Initially η is set at high value and the iterations are carried till the error is within 0.1 times the present threshold. The learning rate is then scaled down by $\{\text{Error}\}^{0.1}$ (chosen heuristically) of its previous value. This process recurs till the present error is less than the threshold. The learning rate is again increased by 1.5 times and the iteration is repeated. If the error again starts decreasing, η is set to 90% of initial value and iterated till a situation is achieved when when the error does not decrease on increment of η.

Table 1 shows the feature value used for training the system. The data is collected from real life patients. The network takes these feature values to train the system corresponding to the given symptoms. Weight matrix, thus formed, was used for testing spectrograms obtained from the Doppler machine. The system perfectly classified the ischemic, normal, vasodilatation and stenosed episodes. Some of the results are shown in the final section (please refer to table 2).

Table 1. Normalized value of features for region left posterior tibial artery. (AS = Acceleration slope, DS= Deceleration slope, NP = Negative peak, SBI = spectral broadening index, A/B = systolic to diastolic ratio, P = Period, SW= Spectral Window, C = Curvature)

Symp	No. of sub	AS	DS	NP	SBI	A/B	P	SW	C
Distal	10	1	1	1	0.5	1	0.94	0.4	1
Vasodln	7	0.14	0.12	0	0.82	0.14	0.90	0.9	0.3
Ischemic	8	0.03	0.01	0	1	0.04	1	1	0.2
Normal	15	0.25	0.23	1	0.83	0.74	0.87	0.67	0.39

3 End Diagnosis

Using the neural network system, the symptom pertaining to the available spectrogram at any particular region is diagonised. The exact arterial status cannot be described from the conclusion drawn from one spectrogram. For a brief explanation of this matter, we take the carotid arterial region (Fig. [3]). For a constriction at the common carotid artery (CCA), effects will be seen on all the branches coming out

from the CCA (please refer to Fig. [3]). So, if a study is conducted over all the branches, a conclusion can be drawn over the status of CCA. If we look at these aspect somewhat closer, we can see the relevance of scanning the whole arterial tree. Let us make a top-down approach. If spectrogram obtained from left external carotid artery shows stenosis then the possible set of inferences are that (a) due to stenosis at the left external carotid artery itself, (b) due to stenosis at left carotid artery, (c) due to that at common carotid artery or (d) due to some mechanical errors. If the left internal carotid artery shows the same symptom, chances of stenosis are there either at CCA or at left carotid artery. If further investigations on the right external or internal carotid show the same symptom, the probability of occurrence of problem is more in the CCA. Thus, the tree scanning is required to make a diagnostic conclusion.

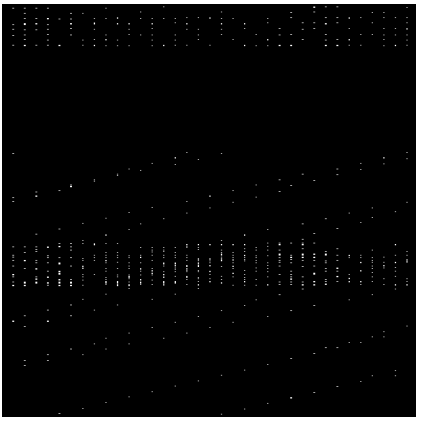


Fig 3. Arterial Distribution within a human body

Thus, a series of spectrograms from different but adjacent regions are studied and the end conclusive diagnosis is drawn based on the findings of all the regions. The goal for this problem-solving system is to collect evidence as the system goes along and to modify its behaviour on the basis of the evidence. To model this behaviour, the Bayesian statistics has been employed, since it is a conditional probability statistics.

Bayesian probabilists interpret probability as a subjective state of knowledge. In practice they use relative frequencies (subsethood degrees) but only to approximate these `states of knowledge'. The conditional probability is given by $P(H \mid E)$.

The expression can be read as the probability of hypothesis H given that the evidence E is observed. Bayesian approach uses all and only the available uncertainty information in the description of uncertain phenomena. This stems from the Bayes' theorem expansion of the `a posterior' conditional probability $P(H_i \mid E)$, the probability that H_i, the i^{th} of the k-many disjoint hypothesis $\{H_j\}$, is true given the observed evidence E:

$$P(H_i \mid E) = \frac{P(E \mid H_i).P(H_i)}{\sum_{n=1}^{k} P(E \mid H_n).P(H_n)} \quad (10)$$

since the hypotheses partition the sample space
X: $H_1 \cup H_2 \cup .... \cup H_k = X$ and
$H_i \cap H_j = \Phi$ if $i \leq j$
In Eq. (10)
$P(H_i \mid E)$ = probability that hypothesis H_j is true given evidence E
$P(E \mid H_i)$ = probability density of E given the hypothesis H_i.
$P(H_i)$ = a priori probability that hypothesis i is true in absence of any specific evidence.
K = the number of possible hypotheses.
Here the probability is based on the previous findings and the hypothesis is set for each of the symptoms. The intersection of the set of findings (in terms of symptoms) leads to the final measure of belief (MB).

The functions should satisfy the following conditions :

- Since the order in which evidence is collected is arbitrary, the combining functions should be commutative and associative.
- Until certainty is reached, additional confirming evidence should increase MB.
- If uncertain inferences are chained together, the result should be less certain than either of these inferences alone.

We define two components :

- MB [H, E] – It is a measure (between 0 and 1) of belief in hypothesis H given the evidence E. MB measures the extent to which the evidence supports the hypothesis. It is zero if the evidence fails to support the hypothesis.

- MD [H, E] – It is a measure (between 0 and 1) of disbelief in hypothesis H given the evidence E. MD measures the extent to which the evidence supports the negation of the hypothesis. It is zero if the evidence supports the hypothesis.

Having accepted the desirability of the properties stated earlier, given two observations E_1 and E_2, the MB and MD of a hypothesis are computed from :

$MB [H,E_1 \Lambda E_2] = 0$ if $MD[H,E_1 \Lambda E_2] = 1$

$=MB [H, E_1]+MB[H,E_2].$

$(1-MB[H,E_1])$ otherwise (11a)

$MD[H,E_1 \Lambda E_2] = 0$ if $MB[H,E_1 \Lambda E_2] = 1$

$=MD[H,E_1]+MD[H,E_2].$

$(1-MD[H,E_1])$ otherwise (11b)

One way of defining MB is :

MB[H,E] = 1 if P(H) = 1

$$= \frac{\max [P(H \mid E), P(H)] - P(H)}{(1 - P(H)).P(H \mid E)} \quad \text{otherwise} \tag{12}$$

Thus, here we search for the maximum probability and the end diagnosis is done based on the results that have the maximum probability when the joint probability is calculated.

The hypotheses are set beforehand and when the symptoms are obtained, each symptom of a certain region corresponds to a set of diseases. The probability matrix gives the chance of occurrence of a problem corresponding to specified symptom in a given region, i.e. the region of probing. When the whole set of observations have been taken, the hypothesis having the maximum probability is taken as the end result, i.e., it has the maximum belief level.

The disease to symptom probability matrix is of the form as shown below:

$$\begin{bmatrix} \text{Prob of Disease} & P(D_1) & \ldots\ldots\ldots & P(D_n) \\ \text{Symptoms} & & & \\ \text{Ischemic} & x_{11} & \ldots\ldots\ldots & x_{1n} \\ \text{Distal} & x_{21} & \ldots\ldots\ldots & x_{2n} \\ \ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots \\ \text{Vasodilata tion} & x_{m1} & \ldots\ldots\ldots & x_{mn} \end{bmatrix} \tag{13}$$

where $P(D_i)$ denotes the probability of occurrence of the disease. So, the matrix gives the probability of occurrence of a disease corresponding to the symptoms. The apriori probability is set according to the statistical data obtained for different age group, region and food habit. This apriori probability is totally a statistical data and can be modified with every new data.

Thus, if region x_1 has symptom S_1, x_2 has symptom S_2, and so on the belief measure for occurrence of all the diseases is calculated. The disease having maximum MB is taken as the inference for the given set of data.

4 Result and Discussion

Spectrogram data has been classified using the BPNN. Four patterns were used for training the network for classification. The representative patterns are normal, distal, vasodilatation and ischemic. Trained networks has been used to identify patterns and used for spectrograms obtained from clinical laboratories.

Fig. 4. Spectrogram corresponding to ischemic flow

Some of the data tested are shown in table 2.

The Fig. 4 shows a spectrogram for ischemic flow characterised by low slopes, blunt curvature and spectral broadening. The probability table used for determining the end result is shown for lower limb (left popletial tibial artery).

Table 2. Result of tested data (the number within braces shows the amount of resemblance)

AS	DS	NP	SBI	A/B	P	SW	C	Result
0.9	0.95	1	0.43	0.87	0.73	0.2	0.87	Distal (0.9)
0.20	0.19	0	0.73	0.09	0.81	0.88	0.22	Vasodln(0.8)
0.03	0.01	0	0.86	0.06	0.92	0.91	0.17	Ischemic (0.9)

Table 3. Probability distribution table (partial picture of the whole range) of different diseases with symptoms in the region LPTA (refer to Eq. (13))

LPTA (Left Popletial Tibial Artery)				
	Femoral Popletial Block	Bilateral Damp Flow	Narrowing , Popletial	Normal and broadening
Normal	0.036501	0.100833	0.170641	0.418624
Distal	0.371210	0.387755	0.105433	0.110240
Ischem	0.570302	0.405234	0.361901	0.160936
Vasdln	0.021987	0.106178	0.362025	0.310310

Using equation (11) and (12) for a set of data the results were seen to be as :

Table 4. Symptoms leading to different levels of MB are shown for two set of tested data in the lower limb region

Region	Lt. Art. Pop	Rt.Art Pop	RPTA	LPTA	Infer
Sympt-oms	Ischemic (0.69)	Distal (0.86)	Distal (0.78)	Ischemic (0.80)	Femoral Pop Block (MB = 0.861034)
	Normal (0.68)	Normal (0.54)	Damped (0.72)	Vasodln (0.76)	DiffusedNarrowing Popletial Downstream (MB=0.71)

The survey shows a tendency of femoral popletial block with subsequent detection of ischemic episode and distal stenotic symptom in the adjacent arteries (Table 4).

5 Conclusion

This paper proposes an automated method to diagnose a disease from a set of spectrograms available from a person's certain region. The feature based extraction takes care of the whole spectrogram while reducing the vector space. The ANN scheme automatically classifies the patterns, and the probabilistic method gives the end result from a number of spectrograms. The main advantage of this automated diagnostic scheme is its use with the system that doesn't have the imaging facility. So even without imaging the system can work as good so as to provide the practitioner with correct interpretation combining the several features it automatically extracted. The system also has the potentiality to adapt to new patterns and add them to the system for future recognition process. When applied to 126 spectrograms from the lower limb the model was seen to correctly interpret for most of the cases. For the feature based classification the result was correct for all observations. For the final interpretation part it showed considerable success (above 80%). Addition of more data, i.e. enrichment of the database would make the system more accurate, since here the estimation is made on statistical basis.

References

1. E.G.M. Petrakis and C. Faloutsos, "Similarity Searching in Medical Image Databases," IEEE Trans on Knowledge and Data Engg., vol 9, pp. 435-447, 1997.
2. H. Fukuda, M. Ebara, A. Kobayashi, N. Suguira, H. Saisho, F. Kondo, S. Yoshino and T. Yahagi, "An Image Analyzing System Using an Artificial Neural Network for Evaluation of the Parenchymal Echo Pattern of Cirrhotic Liver and Chronic Hepatitis," IEEE Trans. on BME, vol 45 (3), pp. 396-399, 1998.
3. E. Chen, P.C. Chung, C.L. Chen, H.M. Tsai and C.I. Chang, "An Automatic Diagnostic System for CT Liver Image Classification," IEEE Trans. on BME, vol 45 (6), pp. 783-793, 1998.
4. C.B. Burckhardt, "Comparison Between Spectrum and Time Interval Histogram of Ultrasound Doppler Signals," Ultrasound in Medicine and Biology., vol – 7 , pp. 79-82
5. A. Caprihan, J.G. Davids, E.R. Greene, J.A. Loeppky and M.W. Eldridge, "Waveform Analysis of Doppler Ultrasound Signals by Microcomputers," IEEE Trans. on Biomedical Engineering, vol 29, pp. 138-142, 1982
6. D.H. Evans and A. Caprihan, "The Application of Classification Techniques to Biomedical Data, with Particular Reference to Ultrasonic Doppler Blood Velocity Waveforms," IEEE Trans. on BME, vol 32, pp. 301-311, 1985
7. F.M. Greene, K.Beach, D.E. Strandness, G. Feli and D.J. Phillips, "Computer Based Pattern Recognition of Carotid Arterial Disease Using Pulsed Doppler Ultrasound," Ultrasound in Medicine and Biology, vol 8, pp. 161-176, 1982
8. P.E. Zuech, R.S.C. Cobbold, K.W. Johnston and M. Kassam, "Spectral Analysis of Doppler Flow Velocity Signals: Assessment of Objectives, Methods and Interpretation," Annals of Biomedical Engineering, vol 12, pp. 103-116, 1984.
9. P.M. Brown, K.W. Johnston, M. Kassam and R.S.C. Cobbold, "A Critical Study of Ultrasound Doppler Spectral Analysis for Detecting Carotid Disease," Ultrasound in Med. and Biol, vol 8, pp. 515-523, 1982.
10. R. Skidmore and J.P. Woodcock, "Physiological Interpretation of Doppler Shift Waveforms - I," Ultrasound in Med. and Biol., vol 6, pp. 7-10, 1980.

11. R. Skidmore and J.P. Woodcock, "Physiological Interpretation of Doppler Shift Waveforms - II," Ultrasound in Med. and Biol., vol 6, pp. 219-225, 1980.
12. R. Skidmore, J.P. Woodcock, P.N.T. Wells, D. Bird and R.N. Baird, "Physiological Interpretation of Doppler Shift Waveforms - III," Ultrasound in Med. and Biol., vol 6, pp. 227-231, 1980.
13. M.S. Kassam, R.S.C. Cobbold, K.W. Johnston and C.W. Graham, "Method for Estimating the Doppler Mean Velocity Waveforms," Ultrasound in Med. and Biol., vol 8, pp. 537-544, 1982.
14. P.G. Kalman, K.W. Johnston, P. Zuech, M. Kassam and K. Poots, "In Vitro Comparison of Alternative Methods for Quantifying the Severity of Doppler Spectral Broadening for the Diagnosis of Carotid Arterial Occlusive Disease," Ultrasound in Med. and Biol., vol 11, pp. 435-440, 1985
15. V.L. Babikian and L.R. Wechsler, "Transcranial Doppler Ultrasonography," Mosby Publications, 1993.
16. S.L. Hagen-Ansert, "Textbook of Diagnostic Ultrasonography," Mosby Publications, 1995.
17. K.J.W. Taylor, P.N. Burns and P.N.T. Wells, "Clinical Applications of Doppler Ultrasound," Raven Press, 1997

Using Randomized Algorithms for Digital Mock-Up in Automotive Industry

Benedikt Bietzker[1], Oliver Karch[2], and Hartmut Noltemeier[2]

[1] ISA Tools GmbH,
Vaihinger Straße 169, 70567 Stuttgart, Germany
bietzker@isa.de
[2] Department of Computer Science I, University of Würzburg,
Am Hubland, 97074 Würzburg, Germany
(karch | noltemei)@informatik.uni-wuerzburg.de

Abstract. An automobile consists of a large number of component parts that must be assembled. Even if all parts precisely fit together, it is not clear whether they can be assembled or not. The process of finding a suitable assembly sequence, which can be performed in reality, is called assembly planning.
We present our probabilistic motion planner RAMONA developed in cooperation with Audi AG, Germany, which is used within a digital mock-up project for checking the feasibility of assembly sequences. The heart of RAMONA is a probabilistic complete motion planner, together with an efficient local path planner. We describe the basic concepts of our algorithm and investigate some details of the local planner.

Keywords: assembly planning, motion planning, randomized algorithm, adaptive search, digital mock-up

1 Introduction

An automobile consists of a large number of component parts that must be assembled. Even if all parts precisely fit together, it is not clear whether they can be assembled or not. For example, it may happen that some part A blocks the installation of another part B and conversely part B is in the way of part A. In addition, there must not only be enough room for the individual parts but also for the necessary assembly tools like robot arms, grippers, etc. The process of finding such a suitable *assembly sequence* (or *assembly plan*), which can be performed in reality, is called *assembly planning*.

Currently, assembly planning is done by hand in an iterative fashion: a mock-up is built (on the original scale of the automobile parts) and used for checking the assembly plan. The mounting sequence of the parts, the way the parts are gripped and fixed, and the shape of the parts are modified until a suitable assembly plan is found. Figure 1 shows as an example some typical assembly tasks in automotive industry. If this process is simulated on a computer, we call it *digital mock-up*. This way we need not build expensive mock-ups and modifications of

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 417–425, 2000.

Fig. 1. Typical assembly tasks performed by various KUKA robots: insertion of front seats (left) and wiring harnesses (right) into the car body

the parts can be directly applied to their CAD models. As a consequence, the whole process of assembly planning becomes faster and cheaper.

The planning department of Audi AG, Germany, currently also focuses on digital mock-up: they are working on a system that can simulate component parts and that is able to decide whether a component can be moved to its mounting position without collisions. To perform the second task we have developed a motion planner in cooperation with Audi AG that is able to find collision-free paths in scenes with a high number of obstacles (i. e., the components of the car) and robots with many degrees of freedom. The objective of our motion planner RAMONA (Randomized Algorithm for Motion and Assembly Planning) is to compute *one* feasible path of arbitrary quality if such a path exists. Here, the quality of the path (e. g., its length) is not very important, since for checking the feasibility of an assembly plan we only want to know whether a collision-free path exists or not.

In the following section we will introduce the general motion planning problem, show that this problem is NP-hard and investigate approximate methods for solving it. In Sect. 3 we will present the basic concepts of our probabilistic motion planner RAMONA and investigate some details of RAMONA in Sect. 4 and 5, the local planner and the collision check algorithm.

2 The General Motion Planning Problem

In general, the motion planning problem can be formally described as follows: a robot living in a physical world, which is modeled by an Euclidean smooth manifold called the *workspace* of the robot, consists of one or more rigid bodies (or *links*) that are attached to each other. A configuration of the robot in its workspace (i. e., the position and orientation of the robot and of all of its

links) is uniquely represented by a vector of parameters, that is, by a point in the *configuration space* $\mathcal{C}$ of the robot. Consequently, the number of degrees of freedom of the robot determines the dimension of its configuration space. The subset $\mathcal{F} \subset \mathcal{C}$ of the configuration space that represents robot configurations not colliding with any obstacles in the workspace is called the *free space* of the robot.

Reif has already shown [9] that the general motion planning problem, where the number f of degrees of freedom is part of the input, is PSPACE-hard and therefore at least NP-hard. Furthermore, the complexity of the free space of the robot was shown to be in $\Omega(n^f)$ in the worst case [3], where n describes the complexity of the workspace, i. e., the total complexity of the robot and all obstacles. The consequence is that *exact* motion planning algorithms relying on the computation of the free space can be quite expensive, in particular for robots with many degrees of freedom.

Therefore, it seems to be meaningful to consider also *approximate* methods that may be more efficient and are often easier to implement than exact algorithms. The price we have to pay for this may be the loss of *completeness*: the algorithm may fail to find a collision-free path, even if such a path exists. But the situation becomes better if we use an algorithm that is at least *probabilistic complete*. We call a motion planning algorithm probabilistic complete if for arbitrary given start and goal configurations $c_s, c_g \in \mathcal{F}$, which can be connected by a collision-free path of the robot in the workspace, the probability that the algorithm finds such a collision-free path converges to 1 with increasing running time. Such an algorithm, together with an efficient local path planner, is the heart of our motion planner RAMONA and will be described in the following sections.

3 The Basic Concepts of Ramona

Generally, RAMONA is a roadmap-based motion planning algorithm as described for example in [5]: in a learning phase a *probabilistic roadmap*, which represents the free space $\mathcal{F}$, is constructed and stored as a graph $\mathcal{G}$, the *connectivity graph*. The nodes of $\mathcal{G}$ correspond to collision-free configurations and its edges represent feasible paths between the configurations. These paths are computed using a simple and efficient (deterministic) local planner (described in Sect. 4). In the query phase, the start and the goal configurations $c_s, c_g \in \mathcal{F}$ are connected to two nodes of the roadmap (again using a local planner); then the roadmap (or the connectivity graph $\mathcal{G}$, respectively) is searched for a path joining these two nodes.

The nodes of the connectivity graph $\mathcal{G}$ are called *milestones*. In the learning phase we start with an empty set $\mathcal{M} = \varnothing$ and iteratively add new milestones to it: we randomly choose a new configuration $c \in \mathcal{F} \setminus \mathcal{M}$ and try to connect it to one of the configurations of $\mathcal{M}$ using the local planner. If we succeed, we add configuration c as a new milestone to $\mathcal{M}$ and update the set of edges of $\mathcal{G}$ accordingly. This process is repeated until we exceed a given time limit or a bound on the cardinality of $\mathcal{M}$.

Algorithm 1 Find a collision-free path between two robot configurations

Input: Start and goal configurations c_s and c_g
Time bound $t_{\max}$ and cardinality bound $m_{\max}$ on the number of milestones
CAD models of the scene

```
1  Initialize spatial data structures using the CAD models
2  M ← {c_s, c_g}, E ← ∅, G ← (M, E)                    {Initialize connectivity graph}
3  t ← 1
4  while (c_s and c_g are not connected in G) ∧ (t ≤ t_max) ∧ (|M| ≤ m_max)
5    Generate new milestone c
6    N(c) ← neighborhood of c in M
7    for all n ∈ N(c) do                    {Try to connect c to its neighbors in M}]
8      if LocalPlanner(c, n) succeeds then
9        M ← M ∪ {c}, E ← E ∪ {(c, n)}                {Update connectivity graph}
10     end if
11   end for
12   t ← t + 1
13 end while
```

Output: If c_s and c_g are connected in $\mathcal{G}$, a collision-free path between c_s and c_g exists (represented by the path from c_s to c_g in $\mathcal{G}$)

An essential part of the learning phase is the local planner, a simple but very fast motion planning algorithm. For example, a straight forward way for connecting two configurations $c, c' \in \mathcal{F}$ is to connect c and c' by a straight line segment in the configuration space and then to check whether the resulting robot motion is collision-free in the workspace (see Sect. 5). If this is the case, we found a collision-free path from c to c', otherwise the local planner has failed connecting c and c'. Usually, such a local planner is not able to connect configurations that are "too far away" from each other, depending on the shape of the free space $\mathcal{F}$. Consequently, the milestones must be chosen densely enough, such that they can be connected using the local planner. Furthermore, it may be even necessary to locally refine the roadmap in "difficult" regions of the free space by adding additional milestones (see [5,7]).

In our setting, we have combined the learning and the query phase into *one single phase*, that is, we directly start with $\mathcal{M} = \{c_s, c_g\}$ as the initial set of milestones and stop as soon as c_s and c_g are in the same connected component of $\mathcal{G}$, that is, c_s and c_g are connected by a path in $\mathcal{G}$. The reasons for this are that we only intend to check the feasibility of *one single* path, the quality of the computed path is not important to us, and the scene (i. e., the positions of the robot and the obstacles) immediately changes after the query. Therefore, a time consuming learning phase would not be profitable for only one single query.

Algorithm 1 describes the main procedure of RAMONA. Some of the steps of the algorithm have to be specified more precisely:

Step 1 First of all, we have to initialize some spatial data structures that are used for the efficient computation of distances between the robot and the obstacles. These distance computations are used by the local planner (see the following section).

Step 5 In our approach, we compute the milestones uniformly randomly distributed over the configuration space. But instead of computing them one by one when they are required, we compute a relatively small *set of possible milestones* at the beginning of the algorithm. Then, we generate a new milestone by choosing an element of this previously computed set. This has the advantage that we can influence the *order* in which the milestones are taken from this set. For example, in environments with low obstacle density it would be useful to process those milestones first that are near to the straight line segment between c_s and c_g. This often results in a shorter path, less milestones and a better running time. If all possible milestones are consumed, we simply generate a new set of possible milestones.
Of course, there are many other ways to compute the milestones besides this naive but very fast approach, e. g., retracting randomly generated configurations onto the medial axis of the free space [12].

Step 6 It would be too expensive if we try to connect a new milestone c to *all* existing milestones in $\mathcal{M}$ using the local planner. Therefore, we first compute a suitable neighborhood $N(c) \subset \mathcal{M}$ of c in the set of milestones and call the local planner only for this smaller set. The neighborhood is defined using an appropriate distance function $d\colon \mathcal{F}\times\mathcal{F} \to \mathbb{R}_{\geq 0}$ between robot configurations. To actually compute the neighborhood $N(c)$ we perform a range query in the set $\mathcal{M}$ with respect to the distance function d. Using an appropriate spatial data structure (e. g., the Monotonous Bisector Tree described in [10, Chap. 4.3]) this query and the update of the spatial data structure if a new milestone is added in step 9 can be performed efficiently.

4 The Local Planner

The basic idea of our local planner is to explore the free space from the start configuration until the goal configuration is reached or a certain time limit is exceeded. This exploration is performed investigating the *clearance* of the robot: the clearance is defined as the minimum distance (in the workspace) between the robot and the set of obstacles. Using the clearance of the robot and the information about its geometry we are able to compute an interval for each coordinate of the configuration space, such that all robot configurations inside the safety-region formed by these intervals are collision-free (see [11]).

Hence, the exploration of the free space can be performed in the following way: we compute the clearance and the corresponding safety-region of the robot (see Sect. 5) and check whether the goal configuration is covered by this region. If this is not the case, we *expand* the current configuration and choose one or more new configurations from the safety-region to be processed next. One of the advantages of this method is, that the size of the steps the robot takes is not fixed, but depends on the clearance of the robot: the larger the clearance

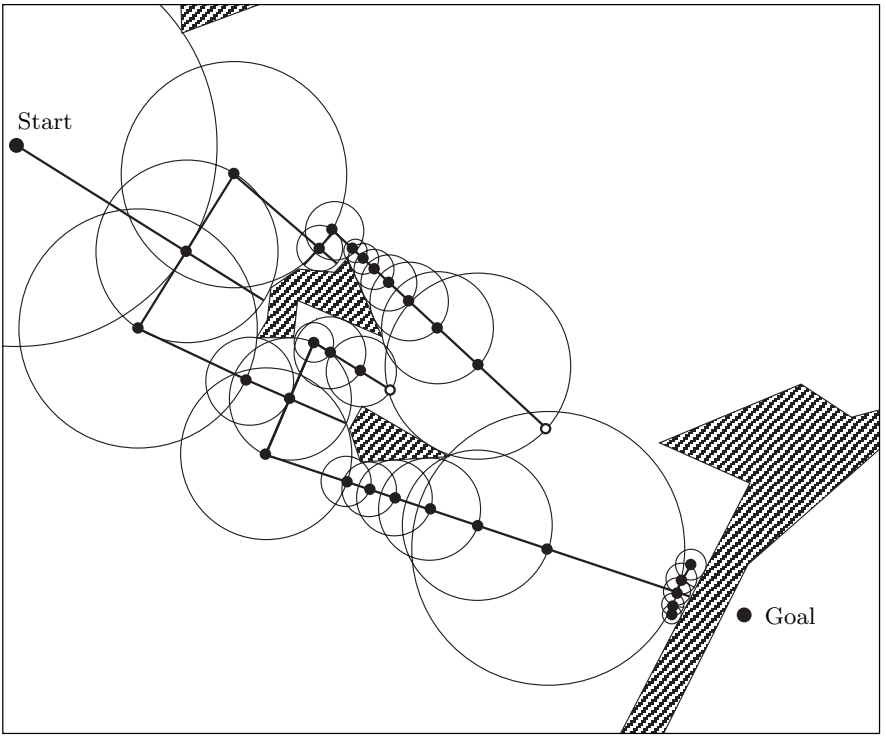


Fig. 2. Example of the local planning strategy in two dimensions

of the robot is, the greater are its steps. So we are not limited to a fixed step size, which often must be carefully chosen by the user. Figure 2 shows a two-dimensional example: the circles represent the safety-region of the robot induced by its clearance.

Furthermore, the search is guided by an A*-algorithm. The value $w(c)$ of the weight function of a configuration c, which controls the order in which expansion steps of the A*-algorithm are performed, both depends on the distance between c and the goal c_g and on the reciprocal of the clearance of c (see [11]). The idea behind is to favor paths directed to the goal configuration that have a large clearance. Then, the robot is able to make large steps and thus can quickly move towards the goal configuration. And since we never enter a collision-free region of an already expanded configuration again, the free space usually is explored very fast.

Of course, as with every local planner it may happen that our planner gets stuck on its way to the goal. Figure 2 shows an example where the way to the goal is blocked by an obstacle: the expanded configurations seem to "converge" against the obstacle. Here, a suitable time limit is necessary that stops the planning algorithm if it takes "too long" finding a collision-free path between start and goal configuration.

5 Collision Check and Clearance Computation

In step 5 of Algorithm 1 (the generation of a new milestone) as well as in the local planner we have to check whether a configuration of the robot is free, that is, does not collide with any obstacle. Both the collision check and the computation of the clearance can be performed by computing the distances between the robot and all obstacles.

Hence, we use an efficient method based on an idea of Quinlan [8] that allows us to compute an approximation of the distance between two non-convex objects very fast. For this, all objects are approximated by small convex objects (e. g., spheres), which are maintained hierarchically. Figure 3 shows an example of such an approximation.

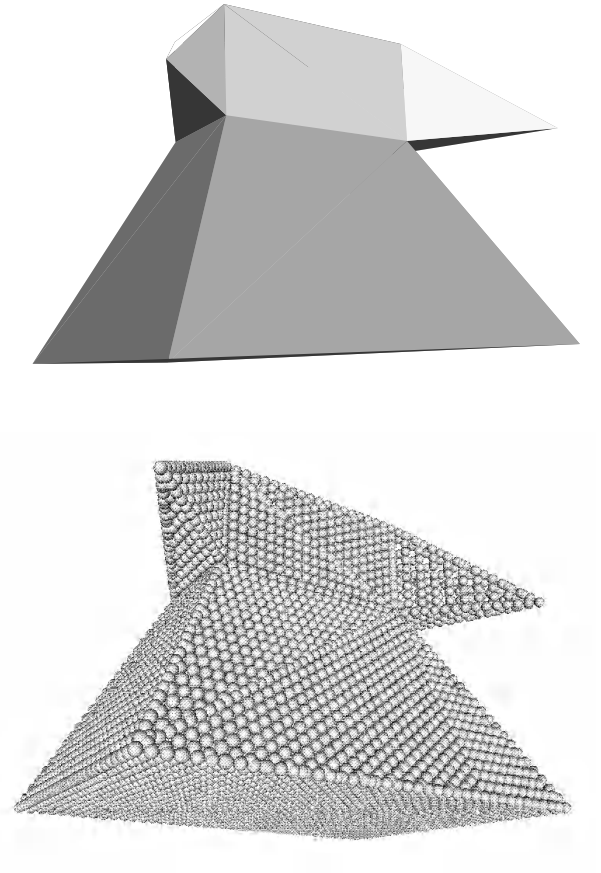


Fig. 3. Approximating an object by the union of spheres

6 Conclusion and Future Work

We presented our probabilistic motion planner RAMONA, which is used for efficiently checking the feasibility of assembly sequences in automotive industry. The heart of RAMONA consists of a combination of a probabilistic complete motion planner with a fast local path planner and a method for efficiently computing distances between arbitrary objects.

In the future we are going to improve several parts of RAMONA independently:

- Currently, we compute a set of milestones uniformly randomly distributed over the configuration space and then consume these milestones one by one. In the future, we will investigate more sophisticated methods to generate the milestones (see [12] as an example) or try to use heuristic algorithms depending on the robot's environment. Also the order in which the milestones are consumed by the local planner can be controlled by a suitable heuristic approach.
- The efficiency of the local planner described in Sect. 4 can also be improved by heuristic approaches, e. g., by using different weight functions that control the order in which the expansion steps of the A*-algorithm are performed. Furthermore, the local planner is also very well suited for an implentation on a parallel computer.

References

1. B. Bietzker. Robot Motion Planning for Digital Mock-up: A Randomized Algorithm for Assembly Planning in Automotive Industry (in German). Master's thesis, University of Würzburg, 1998.
2. R. C. Bolles, H. Bunke, and H. Noltemeier, editors. *Intelligent Robots – Sensing, Modelling, and Planning*. World Scientific, 1997.
3. D. Halperin and M. Sharir. Arrangements and their Applications in Robotics: Recent Developments. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Algorithmic Foundations of Robotics*, pages 495–511. A K Peters, 1995.
4. D. Hsu, J.-C. Latombe, and R. Motwani. Path Planning and Expansive Configuration Spaces. In *Proceedings of the 1st CGC Workshop on Computational Geometry*, 1996.
5. L. E. Kavraki, J.-C. Latombe, M. Overmars, and P. Švestka. Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
6. L. E. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan. Randomized Query Processing in Robot Path Planning. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 353–362, 1995.
7. M. Overmars and P. Švestka. A Probabilistic Learning Approach to Motion Planning. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Algorithmic Foundations of Robotics*, pages 19–37. A K Peters, 1995.
8. S. Quinlan. Efficient Distance Computation between Non-Convex Objects. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3324–3329, 1994.

9. J. H. Reif. Complexity of the Mover's Problem and Generalizations. In *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Sciences*, pages 421–427, 1979.
10. K. Verbarg. *Spatial Data Structures and Motion Planning in Sparse Geometric Scenes*. PhD thesis, University of Würzburg, 1996.
11. K. Verbarg. Planning of Fast and Secure Paths in Complex Configuration Spaces Using Simple Primitives. In [2].
12. Steven A. Wilmarth, Nancy M. Amato, and Peter F. Stiller. Motion Planning for a Rigid Body Using Random Networks on the Medial Axis of the Free Space. In *Proceedings of the 15th Annual ACM Symposium on Computational Geometry*, pages 173–180, 1999.

Risks Inside-Out*

Osvaldo Cairó[1], Julio Barreiro[2], and Francisco Solsona[2]

[1] Computer Science Department
Instituto Tecnológico Autónomo de México (ITAM)
Mexico City, Mexico 01000
cairo@lamport.rhon.itam.mx,
http://cannes.divcom.itam.mx/~osvaldo
[2] Computer Science Department
Universidad Nacional Autónoma de México (UNAM)
Mexico City, Mexico 04510
{barreiro,solsona}@abulafia.fciencias.unam.mx

Abstract. Software project mistakes represent a loss of millions of dollars to thousands of companies all around the world. These software projects that somehow *ran off course* share a common problem: Risks became unmanageable. There are certain number of conjectures we can draw from the high failure rate: Bad management procedures, an inept manager was in charge, managers are not assessing risks, poor or inadequate methodologies where used, etc. Some of them might apply to some cases, or all, or none, is almost impossible to think in absolute terms when a software project is an ad hoc solution to a given problem. Nevertheless, there is an ongoing effort in the knowledge engineering (KE) community to isolate risk factors, and provide remedies for runaway projects, unfortunately, we are not there yet. This work aims to express some general conclusions for risk assessment of software projects, particularly, but not limited to, those involving knowledge acquisition (KA).

1 Introduction

We have heard tales of multi million dollar mistakes in software projects, while the prowess of two students working in a shed built a program superior to that developed by a large professional teamwork at a real low cost. Why does this happen? Are software projects all that risky? Should we replace our professional teams by hackers? Is it simply project mismanagement?.

We know that software projects are notoriously difficult to manage, in particular, those involving knowledge can end in failure pretty easily, since the main problem appears to be due to a lack of methods and tools for knowledge modeling [Cairó, 1998]. Every year new knowledge based systems (KBS) are developed and applied, some of them successfully, but some of them not. Even though we have seen recently an enormous growth in the performance of knowledge engineering (KE) tools and techniques, project failures keep being a common ground in

* This project has been partially funded by CONACyT as project number D.A.J-J002-222-98-16-II-98 (REDII), and Asociación Mexicana de Cultura, A.C.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 426–435, 2000.

press reports. So the problem appears to be not related directly to technology, but to the use or misused of this technology and to the poor understanding of KE methodologies.

In any case, the high failure rate, not only in KBS but on general software projects, is highly related to the concept of risk. Many engineers, professional managers and designers feel that risks are too unlikely to justify expending extra effort and money in dealing with them. Do they know what risk means or do they know the advantages of risk reduction? They sure do, but are not willing to pay the prize of assessing and managing risks. We should re-educate ourselves to see the *extra* resources consumption as an investment and not the other way around.

In the authors opinion the two cornerstones in which every successful software project should be based are:

1. Timely risks identification.
2. Measures to asses and manage these risks.

To accomplish our goal as KBS designers we have to reduce or control risks, hence the first step in that direction is the use of a KE methodology, using a methodology, encourages an appropriate management of our decisions, and leaves on room for controlling the process, control that is essential to bust risks. Most people use the ones that have proved to work well in practice, which is a good start, but not a real solution. The differences among KE methodologies is huge, and most of these differences are subtle but important details, you should try at least to:

- Identify the project's main goals.
- Identify the project's top risks.
- Identify similar projects and analyze the methodologies used by them, if any. This might prove to be a very time-consuming process, because there are several KE methodologies, some of them have lots of documentation, and only a few have references to successful projects developed with their methodology. So you will have problems finding a similar project, understanding the basic similarities and differences of the prospect methodologies, and all this should be done taking into account your own project's goals and risks.

Once you have decided for a particular KE methodology, you should follow it tightly. In spirit, most methodologies follow the Boehm's software risk management plan [Boehm, 1988]. Which –unfortunately– add another blemish to the detection of project's top risks, since the list of risk items proposed by Boehm are not representative of those in typical modern organizations. Recent studies have detected which are the most common risks and their relative importance to project managers [Keil et al., 1998], out of the defense context in which Boehm developed his top ten risk list.

Alas, there is no automatic tool for identifying the project's top risk items nor the right methodology for a particular project. It is the project manager responsibility to detect and rank risks, and strategies. This places top management support as the first (and probably) most important risk every new project

face. Parting from Keil's universal set of risks as base, we have divided up the analysis of common risks in several parts: Section 2 provides a full analysis of project manager desirable skills, Section 3 a ranking of risks into two main divisions: under and out of project manager's control, together with a list of risk mitigation strategies, finally Section 4 offers sound points for project managers to evaluate before committing to the use of a particular KE methodology.

This paper attempts to systematically expose major software projects risks, particularly, but not limited to, those involving KA, since all software projects share basic organization, metrics and management procedures. We also venture giving some rules of thumb to keep these risks under control, most of them supported by recent studies in the risk field.

2 Project Mismanagement

Project mismanagement seems to be the primary cause for the growing number of project failures. In a research study [Cole, 1995] conducted in England among companies that had suffered runaway projects, 54% of these companies tried to employ better management procedures to recover. This implies that either the manager had their attention diverted by too many other activities, or that an inept manager was in charge ([Glass, 1998] offers a detailed analysis of Cole's study). In any case, we should be worried about such big figures: More that fifty percent of these runaway projects were partially due to project mismanagement. This confirms the fact that software management is a complex issue. This also invite us to review current management procedures, topic that is not addressed by any KE methodology, since they are used when a project manager, and even the development team are in position, as would be the case in software development companies with a fixed staff.

It is extremely important in order to minimize risks, that the project manager have the following skills/characteristics:

Knowledgeable. KBS cover the full range of human knowledge domain, and expecting the project manager to be knowledgeable about all of them is impossible. Though we should expect the project manager to comfortably manage all system development areas: requirements analysis, design and modeling techniques, formal specification tools, development tools and languages, etc. That is, it is highly desirable to have a person well versed in Computer Sciences (CS). Unfortunately, only in the U.S. the number of undergraduate degrees in CS is declining (a 43% dropped from 1986 to 1994 is mentioned in [Zadeh, 1998]), while the number of software projects grows at a very large rate every year. This can only mean that more and more software projects are being managed by people unversed in CS, which may lead to the underestimation of crucial parts of the system.

Political. To have a viable project, user commitment is a *must have*. And this can only be achieved through good manager-user relationships. This will help mitigate the risk of creating a product that nobody use, or a product that does not fulfill the requirements or expectations of our users.

Trustworthy. Once we have establish user commitment to the project, we must maintain it. The only reliable way of doing that is by meeting commitments.

Open-minded. It is crucial for the project manager to know and understand what is building. And to ensure the project success is a good idea to let users drive it. This reduce the risk of having the product rejected since it gives users a sense of ownership. On the other hand, it poses the problem of ambiguity and sudden changes, particularly in the application layer of the project, and again, is left to the project manager's ability to address these issues in a timely manner.

Fulfilling this list of *desirable* skills is, by no means, an easy task, and as if this were not enough, Zadeh adds: *..., in the computer industry, the working environment is the environment of cutthroat competition*, which is –unfortunately– truth and companies put stock price above human welfare, phenomenon that has pushed computer scientists efficiency higher every time. Hence, projects managers must have a good deal of patience, know stress managing techniques, and have direction skills, since it is precisely on them that the overall management of the system rests.

In [Keil et al., 1998] regarding project managers, is also pointed the imminent danger if they overestimate their own abilities or fail to realize their own limitations for that can lead to an underestimation of risks or a failure to develop appropriate risk mitigation strategies.

It is worth noting that in this discussion the role played by users, is a generic one, since the project manager has to deal with human experts (HE), found sponsors (FS) or company senior management, and finally with customers/users. If either human expert, top management, or user commitment is lacking, this might represent the greatest threat to the project's success. We cannot stress enough the importance of this relationship building.

One of the biggest problems in KBS construction is the acquisition of knowledge from HE, this problem has two faces. On the one hand, we don't really know how to represent the implicit, detailed knowledge of HE. Even though much is known about the biochemical activity of the brain, little is known about the way memory and reasoning work, knowledge is too rich: extensive, inaccurate, incomplete, etc. [Cairó, 1998], to be transfered directly from the different knowledge sources to artificial machines. On the other hand, representing knowledge is but just the final part of a complex process, in which HE tend to see the project itself as a threat to their way of life, as something that will replace them. HE will not cooperate in the process if they are not actively involved in the requirements and goals adjustments of the project. Again, a good and solid relationship between the project manager and HE can save the situation. Note that this particular relationship is necessary, but not sufficient. If inadequate KA techniques, or knowledge modeling methods are applied, not even full HE cooperation will give satisfactory results.

3 Measures to Asses and Manage Risks

In the previous section we focused on the role of the project manager as a key to extenuate some of the most important threats to a successful KBS project. An skillful project manager is, by all means, desirable. Nevertheless, a KBS range from middle to large scale projects, from normal to mission or safety critical, thus a handful or even hundreds of people might be involved with a particular KBS through its life cycle. In this context, the project manager plays the role of *stage director*, he is in charge of supervise and direct the action during the development of the system, however a large scale project has so many important details, that a project manager might, at most, point them out, and have somebody else taking care of them, involvement of the whole staff is crucial. So how can we detect risks, and how should we face them?

Even having a competent project manager that can ruled out the most common cause of software project failures, namely top management, and user commitment, we are left with the task to pinpoint every single problem that may lead the project to a dead end. Some of the most important are:

- Failure in eliciting or specifying customer requirements, having requirements that are unmeasurable, which will prevent us from determining whether a given solution mets that requirement.
- Inappropriate staffing to successfully build the KBS.
- A poor development methodology, or a methodology unsuited for the system.
- Lack of (or poor) time and cost estimations.
- Failure to detect potential knowledge sources, and lack of tools or techniques to elicit and model the knowledge available from such sources.
- External factors, like outsourcing faults or delays, change in the scope of the project, time pressures due to changes in the competitive environment, etc.

The number of things that can go wrong is unnumbered, the key to succeed is to detect such things in a timely manner, and have a team in place that can undertake such problems. In [Keil et al., 1998] several project managers identify a universal set of risk factors, and they proposed a framework for thinking about four distinct types of software project risk, their classification overthrows that proposed by Boehm. We will take the middle ground and divide the risk spectrum into two basic categories: Those that are under direct control of the project manager and those over which the project manager has little or no control. This gives us the option to add risks that exceeds the borders in Keil's framework, and are thus difficult to place in a particular type of risk.

3.1 Risks under Direct Control of the Project Manager

These kind of risks are those involved with the scope, limitations, requirements analysis, proper estimations, knowledge elicitation, modeling, software development, etc. That is, the things that project managers, knowledge engineers, designers, and programmers do, and over which the project manager has, in general, a good level of control.

Faults introduced during the phase of requirements analysis, can lead the team to produce a system that customers did not ordered, hence a system nobody will use. In safety critical systems, things are even worst, because these failures may cause accidents. To mitigate the risk pose by ambiguities while establishing the project's scope and limitations, is a good idea to rely on an evolutionary approach to the specification of software requirements. Most KE methodologies nowadays support an spiral-like life cycle model, which among other things allow requirements clarification and gives a more realistic view of what the system will do. Predictive quality models [Lanubile and Visaggio, 1996] are useful for predicting the quality of software components early in the software life cycle, which means that we can know after the design phase (or early coding phases) which parts are more subject to fail, and so project managers can afford extra resources consumption on inspecting or testing those error-prone components to see if they meet the requirements.

Project managers should consider, for safety critical KBS, to apply detailed safety analysis [Lemos et al., 1996] to determine the risk associated with requirements specification and assess whether this is acceptable within the context of system risk. Safety analysis of software requirements specification gives a rough estimation of the overall system risk, contrary to the belief that safety analysis can only be applied to software (because *safety* is an attribute of the system rather than software.) If the risk is not acceptable, then the specification has to be modified or extra precautions must be consider in order to reduce the risk.

The system development and implementation also fall into the project manager's domain of control, it requires a deep understanding of the requirements and scope of the project though. To experienced project managers, as pointed out by Keil, these kind of risks are regarded as moderate opposed to high level risks, as implied by risk regular literature –as in [Boehm and Ross, 1989]–, since these kind of risks are the only ones covered. It is not our intention to insinuate that these risks are unimportant, on the contrary, if the project manager does not follow a well supported software methodology, or fails to assess support and commitment from project's development team things might go ill: System will be delivered late (if delivered at all), it will not satisfy the requirements, or at the very least will consume more resources (time, money, people, outsourcing, etc.) Among possible mitigation strategies to counterattack these risks are the use of a good set of metrics and evaluation techniques, both internal and external, tools to estimate time and costs, and a clear division of responsibilities among the development staff.

3.2 Risks out of Project Manager's Control

The risks that are outside the range of the project manager control are lack of top management, lack of user commitment, and environmental risks (like conflicts among different user departments, changes in the scope and/or general project goals, markets shift, etc.). The first two, as mentioned before in Section 2, when we were talking about project mismanagement, are crucial to any software project. They depend on an skillful project manager who should be able to

promote, achieve, and maintain good relationships with all the different groups involved in the KBS. We talk lengthly about this particular set of risk in the aforementioned section, because they are tightly related to the top management of the project, and project managers themselves are the best risk mitigation strategy for these kind of risks.

The last kind of risks, environmental ones, have a very low probability of occurring, because they owe themselves to –normally– sudden and unexpected changes in business management, employers cutting down financial support, changes in internal policies that may affect the original goals, etc. They are hard to anticipate, and hence mitigation strategies are but just vague ideas to what people had suggested in the past: contingency, and disaster planning. When one of these risks strikes, the system ends up in a very inconsistent state, because changing the scopes or goals of an ongoing KBS, usually implies major changes to the system requirements, which produces a chain reaction that affects almost every part of the system. Most of these projects are abandoned or otherwise the cost and time doubles, and the risk of having more bugs during the implementation phase grows at least an order of magnitude due to inherited models and code unmodified from the previous version of the KBS.

Managerial roles, organizational design, and business practices are not normally taken into account when trying to mitigate risks. From the manager's perspective, they are –usually– considered as *fixed*. Fortunately, this has started to change. Highly successful Japanese companies (e.g. Honda, Canon, Matsushita, NEC, Sharp, etc.) have become famous for their ability to adapt quickly to customers' needs, developing new products to fulfill those needs, creating, in most cases, new markets.

[Nonaka and Takeuchi, 1995] stated that the centerpiece of the Japanese approach is recognizing that creating new knowledge is not simply a matter of processing objective information, but to dip into the tacit and often highly subjective insights, intuitions, and hunches of individual employees and making those insights available for testing and use by the company as a whole. Once again, the key to this process is personal commitment, the team members' sense of identity with the enterprise and the mission. This is the main topic of knowledge management and organization, and so, it is out of the reach of this paper. However, this approach should not be ruled out as an effective mitigation risk strategy, only that it works over the organization itself, instead of over a particular KBS.

4 KE Methodologies

We now know that through the use of a methodology we can gain finer control over the risks in a KE engineering project, though the myriad of current KE methodologies poses a difficult obstacle to system designers, project managers, and in general to all the crew involved in the development process of a KBS.

Depending on the problem at hand, the project manager should select a methodology taking into account several vital components of the methodology

itself, for instance: automatic knowledge acquisition, ease of project management, modeling environments, fast prototyping, documentation, and *glue mechanisms* (that is interfaces or protocols to connect external management, evaluation, knowledge attain/transfer, validation, and verification tools to the main project's stream). On-line Information, and in general literature about these issues is usually available in the form of papers, technical reports, or manuals. Unfortunately, something rarely taken into account is the learning curve of the methodology itself: We may know what a particular methodology offers by a *coup d'oeil* of the documentation that comes with it, and thus making an informed decision of whether it fulfills our needs might be stright forward. But how much time will it take us to master the methodology itself, or how easy will it be to add/modify a particular aspect of it?

The lack of standardization in the way different KE methodologies are presented is likely to change in the near future with the tremendous hit that the Unified Modeling Language (UML) is having in the computer industry. Having different KE methodologies using the same notation, would ease: mastering all aspects of the methodologies themselves, drawing similarities and dissimilarities among them, extending them to fill any particular need during their application, etc.

On the other hand, every KBS is a whole different problem in itself, so expecting a methodology to completely cover all the particular requirements every new system might have is inconceivable. Nevertheless, almost all KBS share a common set of aspects to deal with, and most KE methodologies take them into account. Though, we must not forget that KA is a modeling activity, and a modeling activity is a constructive[1] one, and therefore there cannot exist a single correct solution.

4.1 KAMET Embraces UML

KAMET[Cairó et al., 1999] stands for Knowledge Acquisition Methodology, it is a modeling methodology designed to manage KA from multiple knowledge sources. Our goal is an 80%–80% approach: We want KAMET to work for 80% of the applications, and for 80% of the knowledge modeling work. From its outset we have been willing to embrace new ideas, among the current and more important characteristics of our modeling methodology we count the following:

1. Spiral life cycle model. Which provides a strong mechanism to achieve KA in an incremental fashion.
2. Various eliciting techniques tightly integrated.
3. Presented in a cooperative framework that helps all the parties involved in the project to gain a deeper understanding of the models.
4. Described using UML. The four stages have been completely redesigned to use UML.

[1] The constructivist view define human knowledge as a cognitive process for which the problem to be solved becomes an artifact.

KAMET is being fine tuned to assess KA from multiple knowledge sources, the four stages that conform the KAMET life cycle, are incremental refinements of an initial model of the system. Such refinements are all represented in sets of UML diagrams, and we also provide useful indicators of software product quality, mapping the requirements to different points during the development process.

The spiral approach naturally favors an equilibrium between management and knowledge modeling. It allows project managers to categorize and asses the knowledge involved in the project, by maturing the experiences acquired during early phases of development, while giving the means for fast prototyping: placing new ideas into the model, and evaluating their progress at the next revision point during the spiral cycle, making the appropriate adjustments and move on. Finally, the use of an object-oriented approach, such as UML, lessens the KAMET's learning curve, and allows users to easily extend KAMET in any particular way to fit their needs.

Together all these characteristics make of KAMET an excellent choice for project managers, particularly those inexperienced or without previous exposure to risk management.

5 Conclusion

We have well-founded suspicions to assert that the high failure rate in software projects (including KBS) is related directly to the miss handling of risks. Even more, we can even draw general conclusions about why this happens: Technology (KE methodologies and software engineering tools) are poorly understood and in some cases misused, project managers are not taking the necessary precautions to detect and assess risks.

We attack three different aspects of the a regular KBS development process trying at all times to focus on the risks that may arise, and possible strategies to minimize them:

- Project Mismanagement
- Measures to asses and managed risks
- Lack of KE methodologies standardization

These three aspects are closely related, and we cannot afford to let aside any of them. They all should be taken into account at the very outset of system development, or carefully schedule to be considered at critical points during the development process.

This paper has been developed with the conscious aim of offering a checklist for new system designers and project managers, from the very conception of a new KBS, including hints to make more informed decisions while choosing a methodology. We also propose a set of rules of thumb, based mainly on the experience of many expert project managers from several countries, condensed at [Keil et al., 1998]. Note though, that this rules of thumb are of very general application, and serious thought must be giving to any of them before making the investment to apply them to a particular KBS development phase.

Needless to say, there is still a long way to go before we can tell the ultimate word on the risks department. Lots of KE, and software engineering groups around the globe, are conducting studies to test the effectiveness of different strategies to assess risks, and to standardized KE and software methodologies, cost and time estimation tools –which in the presence of knowledge fail miserably, and simply because nowadays modeling technology is too poor to represent the human knowledge in full extent–, quality test models, and in general, all the tools that ease the work of KE and make it harder to miss the mark.

Finally, we have reviewed the most important problems to which all project managers are faced when using KE methodologies, what are their differences, how should one go at picking the most appropriate one for the KBS at hand, and all these without losing from sight the risk inherent to the use of a given methodology. We have highlighted KAMET, our modeling methodology, in which much thought have been placed to ease its usage and general applicability.

References

[Boehm, 1988] Boehm, B. (1988). A spiral model of software development and enhancement. *IEEE Computers*, pages 61–62.

[Boehm and Ross, 1989] Boehm, B. and Ross, R. (1989). Theory-w software project management: principles and examples. *IEEE Trans. Softw. Eng.*, 15(7):902–916.

[Cairó, 1998] Cairó, O. (1998). The kamet methodology: Content, usage, and knowledge modeling. In Gaines, B. and Mussen, M., editors, *Proceedings of the* 11^{th} *Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, pages 1–20. Department of Computer Science, University of Calgary, SRGD Publications.

[Cairó et al., 1999] Cairó, O., Barreiro, J., and Solsona, F. (1999). Software methodologies at risk. In Fensel, D. and Studer, R., editors, *11*th *European Workshop on Knowledge Acquisition, Modeling, and Management*, volume 1621 of *LNAI*, pages 319–324. Springer Verlag.

[Cole, 1995] Cole, A. (1995). Runaway projects –cause and effects. *Software World*, 26(3).

[Glass, 1998] Glass, R. L. (1998). Short-term and long-term remedies for runaway projects. *Communications of the ACM*, 41(7):13–15.

[Keil et al., 1998] Keil, M., Cule, P. E., Lyytinen, K., and Schmidt, R. C. (1998). A framework for identifying software project risks. *Communications of the ACM*, 41(11):76–83.

[Lanubile and Visaggio, 1996] Lanubile, F. and Visaggio, G. (1996). Evaluating predictive quality models derived from software measures: Lessons learned. Technical report, University of Maryland, Computer Science Department, College Park, Maryland 20742.

[Lemos et al., 1996] Lemos, R. d., Saeed, A., and Anderson, T. (1996). On the integration of requirements analysis and safety analysis for safety-critical software. Technical report, University of Newcastle upon Tyne, Department of Computer Science.

[Nonaka and Takeuchi, 1995] Nonaka, I. and Takeuchi, H. (1995). *The Knowledge-Creating Company*. Oxford Univ Press, New York.

[Zadeh, 1998] Zadeh, L. A. (1998). Editorial uc-berkeley computer science commencement address. *IEEE Transactions on Systems, Man, and Cybernetics –Part C: Applications and reviews*, 28(1).

A Specific Domain Translator Application in a Floristic Digital Library

Dulcinea Navarrete, Rogelio Dávila, and Alfredo Sánchez

Centro de Investigación en Tecnologías de la
Información y Automatización - CENTIA
Universidad de las Américas Puebla
72820 Sta Catarina Martir, Puebla, México
{is095892, rdav, alfredo}@mail.udlap.mx

Abstract. In this paper, SAVIA is introduced; a system for attaining automatic translation (English-to-Spanish) of morphological descriptions of plants. It was developed at Universidad de las Américas Puebla, as part of a collaborative research with the Flora of North America (FNA) project, currently under construction at the Missouri Botanical Garden [Schnase et al. 1994]. The system has been put forward as a potential solution to the problem of knowledge distribution and multilingual access to Digital Libraries.
Digital Libraries (DLs) are one of the emerging fields of study in computer science and one of the more interesting multidisciplinary research areas. Knowledge distribution is one of its main objectives; hence, the need to develop multilingual applications that make knowledge available to every community that cannot communicate in the language in which DLs material and services where implemented.
A prototype of the system has been developed using technologies based on Glossary-Based Machine Translation and the semantic grammar model. Some examples are presented showing the possibilities of the system. Conclusions will be drawn from the performance of the system and suggestions are included considering further developments.

Keywords: Machine Translation, Glossary-Based Machine Translation, Digital Libraries, Floristic Digital Libraries, Semantic Grammar, tokens, Morphological Descriptions,Treatments.

1 Introduction

It is well known that survival of human genre depends on our capacity to understand and preserve our natural world. Plants, in particular, play a role in maintaining life on Earth. They are the only beings having the capacity to capture energy from the sun and fixing it into organic molecules; a process called *photosynthesis*. Virtually all life on Earth depends, directly or indirectly, on this process.

Important resources have been provided, from different countries, in order to collect information about plants and making it accessible to different communities around the world. Such is the aim of the huge project refereed as Flora of North America (FNA), currently under construction at the Missouri Botanical Garden [Schnase et al. 1994].

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 436—442, 2000.

FNA will capture and make available information of 20,000 specimens of vascular plants and bryophytes of the continental United States, Canada and Greenland .

The aim of a Digital Library (DL) is to provide global access to a large and organized repository of data and knowledge. Through DLs, people will be able to collaborate with their colleagues across geographic and temporal distances [Schnase et al. 1994]. A Floristic Digital Library (FDL) consists of a large repository of botanical documents and a collection of services, which facilitate use and expansion of knowledge about plants. A FDL represents an effort to make flora information accessible to wider audiences around the world and allows botanical researchers to take advantage of ultimate advances in computer technology. However, there are many technical problems in collecting plant information and making it available for communities of researchers around the world. Plant information has been collected for many years by different cultures, in different languages and from geographically distant regions around the world. The present project is an attempt to overcome part of the problem by showing that machine translation technics can be applied in order to provide a high quality low cost machine translation system which enables the possibility of accessing information stored in a DL using different languages. The project is a very ambitious one that only part of the problem will be considered at this stage. That of translating English morphological descriptions of plants into Spanish .

2 SAVIA: A Morphologic Description Translator

Machine Translation (MT) is a computer application area that aims to the automatic translation of texts from one natural language into another. One purpose of MT is to speed up and facilitate worldwide information dissemination. Fully automated MT is a very difficult and ambitious project, and research in the area is still far away to make it a reality. However, successful systems have been developed in restricted areas with a sufficiently narrow subject domain. This approach to MT has come to be known as the sub-language approach [Nirenburg 1987]. We think morphologic descriptions are an interesting area for application as vocabulary and grammar are restricted to a specific domain (botany terms); and morphologic descriptions have a controlled format with specific values.

There are four entities that are maintained in a FDL which are relevant to the project: taxonomic keys, distribution maps, illustrations and treatments. FNA researchers study plants by examining herbarium and critically evaluating published reports on previous works. Information resulting from analysis of a plant is referred to as a "treatment". Treatments are the most important entity for a flora, as they provide detailed morphologic descriptions of plants. Each morphologic description is a list of their morphologic characteristics known as taxonomic descriptors [Sánchez et al. 1994], [Jones 1986]. A taxonomic specialist reviews and edits each treatment. Figure 1 shows an example of a treatment.

Morphological descriptions are edited in a specific format, however one differs from another because there are many editors who edit in different way a treatment. We noticed that the vocabulary used in morphological descriptions is in many aspects very restricted, that is, it resembles very much a formal language designed for enable the communication among botanists. Based on this feature we considered the possibility of using MT technology in order to translate these treatments to different languages.

PSILOTACEAE
John W. Thieret
1.PSILOTACEAE Kanitz Whisk-fern Family

Plants perennial, terrestrial or epiphytic, with corallike, rhizoid-bearing, branched, subterranean axes. **Roots** absent. **Aerial** shoots simple or dichotomously branched; appendages leaflike or bractlike, alternate to subopposite, veinless or 1-veined, less than 1 cm. **Synangia** globose, of 2--3 fused, homosporous eusporangia, solitary in axils of shoot appendages, dehiscing loculicidally. **Spores** many, reniform, not green. **Gametophytes** subterranean, mycotrophic, fleshy, elongate, and branched.

Genera 2, species 4--8 (1 genus, 1 species in the flora): worldwide in tropical regions.

Fig. 1. Morphological description in a treatment.

The objective of SAVIA is made WWW available translations of morphological descriptions accessible in different languages. When a user enter to FDL, SAVIA analyzes the user profile to know which type of user is accessing the system and then SAVIA decides which language (English or Spanish) to use to present the zmorphological descriptions.

2.1 SAVIA. Architecture

SAVIA´s Prototype includes:

- A Glossary-Based Machine Translation (GBMT) engine that provides an automatic translation for an English-Spanish pair.
- Morphological analyzers, a glossary and the semantic analyzers.
- A database keeping user profiles and morphological descriptions that may be reviewed and corrected by an expert.

A general overview of SAVIA functions is shown in figure 2. The input for SAVIA is a taxonomic treatment that is formatted using HTML code. This treatment is processed to produce the morphologic description. This process is named *pre-edition*. Then we use a GBMT to translate the morphological description into Spanish. The *post-edition* function allows a botanical expert to refine the translation generated by the system.

3 Glossary Based Machine Translation (GBMT)

The Glossary-Based Machine Translation engine is the core component of SAVIA. We use a morphological analyzer (scanner) that receives the morphologic description text to produce a list of tokens using database access. At this stage, translations are added for individual words using the bilingual glossary. Then, a parser recognizes each token using a semantic grammar (figure 3). A semantic grammar is a domain related collection of productions which taken together specify precisely a set of acceptable input sequences in terms of an abstract set of terminal tokens. Every language has restrictions on how words must be arranged to construct a sentence.

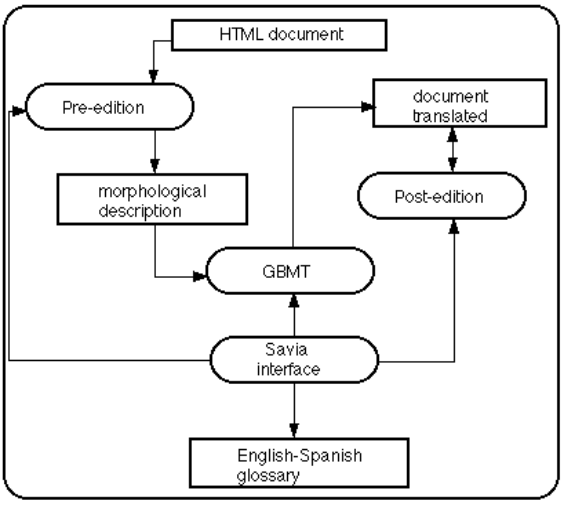


Fig. 2 A simplified overview of SAVIA's basic functions

```
<morf_descrip>:        <morf_descrip> <def_structures>
                       | <def_structures>
<def_structures>:      < name_structures> <descriptions> PUNTO
<name_structure> : STRUCTURE
                       | HABIT STRUCTURE
<descriptions>:         DURATION SIGNO <habit_conj_habit> SIGNO
                       CONJUNCTION APPEREANCE SIGNO HABIT
                        SIGNO HABIT SIGNO LOCATION STRUCTURE
                       | PRESENCE
                       |  ARCHITECTURE  CONJUNCION  HABIT  HABIT  STRUCTURE
<appareance_conj_appareance> SIGNO
<arrangement_prep_arrangement > SIGNO
 <num_veins_conj_num_veins> SIGNO <comparison>
                       | SOLID_SHAPE SIGNO PREPOSICION RANGO ADJETIVO
                  SIGNO ADJETIVO STRUCTURE SIGNO ARRANGEMENT
                        PREPOSICION <structure_prep_structure> STRUCTURE
                        SIGNO HABIT DEHISCENCE
                       | ADJETIVO SIGNO PLANE_SHAPE SIGNO ADVERBIO
                        COLOR
                       | LOCATION SIGNO HABIT SIGNO TEXTURE SIGNO
                        PLANE_SHAPE SIGNO CONJUNCION HABIT
<habit_conj_habit>:HABIT CONJUNCION HABIT
<appareance_conj_appareance>:   APPAREANCE CONJUNCION APPAREANCE
<arrangement_prep_arrangement>: ARRANGEMENT PREPOSICION ARRANGEMENT
<num_veins_conj_num_veins>:NUMBER_OF_VEINS CONJUNCION
                                    NUMBER_OF_VEINS
<structure_prep_structure>:       STRUCTURE PREPOSICION STRUCTURE
<comparison>:          ADVERBIO PREPOSICION MEDICION
```

Fig. 3 Fragment of SAVIA´s semantic grammar

Terminal tokens components of this grammar are botanical terms related to morphological descriptions and syntactic components such as verbs, prepositions, conjunctions, punctuation marks, etc. Botanical terms were taken from the botanical glossary compiled by R. W. Kiger [1996]. An example of the glossary is shown in table 1. Type term refers to the meaning of the word in the morphological description. For example *abaxial* refers to *position*. Non terminal tokens match grammar rules, we analyze every sentence of the morphological description. These sentences begin with the name of a *structure* of the plant (Roots, Leaves, Steam, etc.) followed by a group of *characteristics* to which a value can be associated. An example of the sentence described above is shown in figure 4.

Table 1 Example of SAVIA´s bilingual glossary

English_term	*gender*	*number*	*Spanish_term*	*type_term*
Branched	f	p	ramificadas	habit
Branched	f	s	ramificada	habit
Branched	m	p	ramificados	habit
Branched	m	s	ramificado	habit
Plants	f	p	Plantas	structure
with	w	w	con	conjunction

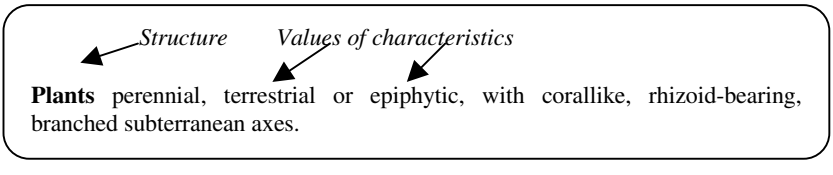


Fig. 4 A sentence of morphological description

4 Implementation

The prototype uses a semantic grammar implemented using Yacc and Lex named *scanner* . The morphological description is extracted from a treatment with a program developed by Abascal [1998]. The grammar is then used to translate the morphological description text into Spanish. The implementation of *scanner* uses database accesses, as tokens are stored in the bilingual glossary. With our semantic grammar, we extract semantic structures from the morphological description. From these structures we extract what we called the semantic representation of the text which consists of a frame-like structure where slots consist in property-value pairs from which the translation into Spanish is obtained. It is important to notice that given this semantic representation it is easy to consider the possibility of extend the system to generate translations of the morphologic structure into different languages.

A fragment of this grammar was shown before in figure 3. Also we use C and CGIC compilers for rapid runs and development. SAVIA provides a translation accessible via World Wide Web. The translation produced by SAVIA of the morphological description example is shown in figure 5.

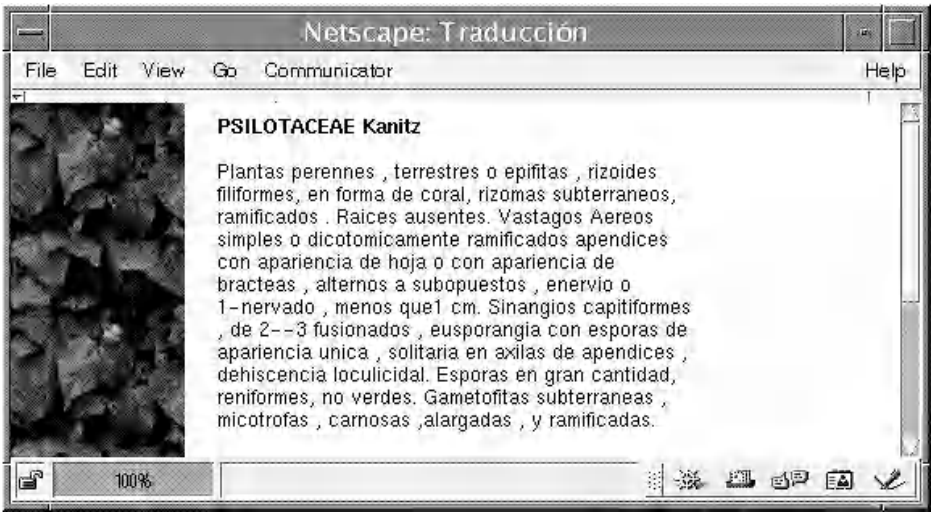


Fig. 5. Example of a Translation made by SAVIA

5 Conclusions

In this paper a system for English-to-Spanish automatic translation of morphological descriptions of plants SAVIA has been introduced. The system was developed as part of the Flora of North America (FNA) project, currently under construction at the Missouri Botanical Garden [Schnase et al. 1994] and has been put forward as a potential solution to the problem of knowledge distribution and multilingual access to Digital Libraries. It is important to realize the potentiality of the system to undertake translation into many languages, different from Spanish, a possible future extension for the system.

It should be noticed that much of the success of the system application comes from the restricted English language used by botanists in the edition of morphological descriptions of plants. The semantic grammar model used for the development of the system, was crucial for attaining high quality translations for the current application, but it has the disadvantage of been a domain dependent model. It means that it would be very difficult to adapt the system to a non-floristic digital library application.

6 References

[Abascal et al., 1994] Abascal R. 1998. X-tract. Tesis de Licenciatura, Departamento de Ingeniería en Sistemas Computacionales,Universidad de las Américas-Puebla. Cholula, Puebla, México.

[Schnase et al., 1994] Schnase, J.L., Legget, J.J., Metcalf, T., Morin, N.R. , Cunnis, E.L., Turner, J.S, Furuta, R.K.,Ellis, L., Pilant, M.S., Ewing, R.E., Hassan, S.W. y Frisse, M.E.,1994. The CoLib project: enabling digital botany for 21st century. *Proceedings of Digital Libraries* 94 (DL '94, College Station, Tex. June).

[Sánchez et al. 1994] Sánchez, J. A., López, C. A., y Schnase, J. L. 1997. Chrysalis: User agents in the construction of floristic digital libraries. *Primer Encuentro de Computación* (ENC 97) (Querétaro, Qro. México, Sept.) ,16-23.

[Jones 1986] Jones, S. 1986. *Sistemática Vegetal.* Mcgraw-Hill, México.

[Niremburg 1987]Nirenburg, S.,1987. *Knowledge and choices in machine translation. In Machine translation: Theoretical and methodological issues*, S. Nirenburg, Ed., Cambridge University Press, Cambridge, Great Britain, 1-21.

A Cooperative, Deductive, and Self-Adaptive Web Authoring Environment

Dominique Decouchant[1] and Ana María Martínez-Enríquez[2]

[1] Laboratoire "Logiciels, Systèmes, Réseaux", Grenoble, France
[2] Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, D. F., México
Dominique.Decouchant@imag.fr, ammartin@mail.cinvestav.mx

Abstract. Related to the cooperative editing research domain and to the works currently developed on the World Wide Web, we present AllianceWeb, an editing and cooperative authoring system on the Web. Using this system, authors distributed among the world can cooperate producing large documentations in a consistent and concerted way. Taking benefits of the design and experiment of the Alliance cooperative editor on Internet, the AllianceWeb approach proposes a mixed architecture (hybrid and/or fully distributed) for the document storage and access. These two architectures are integrated to provide a concerted, secure and parameterizable cooperative editing support.
Following that, we highlight the main aspects of the group awareness function which allows each author to diffuse his contribution to other coauthors, and to control the way by which other contributions are integrated in his environment. In order to support this function, fundamental for every groupware application, techniques of Artificial Intelligence research domain allow to design and to define a self-adaptive cooperative interaction environment, parametrized by user preferences. Thus, the problem and the characteristics of a group awareness inference engine are defined.

Keywords: Cooperative and distributed authoring, World Wide Web, group awareness inference engine, deductive and adaptive cooperation interface.

1 Introduction

The World Wide Web technology enables users distributed over the Internet network to access a growing amount of information. However, the information is stored and managed on Web server sites, and users only remain consumers of information. In this way of operation, Web servers (HTTP daemons) only reply to requests of remote clients (mainly browser applications).

In spite of existence of several interactive browser/editor tools, the information located on WWW servers is usually updated manually by the site manager: documents are prepared in an editing environment and then pushed into the public space on a server. At the same time, many current trends in Web publishing research focus on providing elaborate functions to access documents and publish

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 443–457, 2000.

on a remote side: a user is allowed to get existing documents, modify them, and send them back to the remote server.

In this contribution, before presenting our proposition for a deductive and adaptive cooperative environment for the Web (more precisely we propose a group awareness inference engine), first we need to introduce our approach supporting cooperative authoring, the works currently developed in this area, and finally briefly we introduce our AllianceWeb project.

So, in section 2, we briefly present the two different approaches to support the synchronous cooperative production of large documents : the commonly used *"hypertext"* approach and the less-used but powerful *"document centered"* model. The second solution is chosen highlighting the importance and the gain implied by document structuring.

Then, in section 3 we outline the current state of the art in the domain of Web cooperative authoring and analyzes the main features of the WebDAV (Web Distributed Authoring Versioning) project [16]. Relatively to this proposition, we present main goals and concepts of the AllianceWeb project [5] whose the aim is to define and provide a powerful environment to support cooperative writing of Web documentation.

Finally in section 4, relatively to the innovative features of the AllianceWeb project, we introduce the first elements to design and to provide a deductive and adaptive support of the group awareness function [6][14] for this elaborated cooperative Web authoring environment.

The design of the group awareness function is the kernel of any groupware applications that requires important attention. Thus, we propose to take benefits of the know-how of the Artificial Intelligence domain to design a deductive and adaptive group awareness inference engine. In regards to actions performed by the user and analyzing his choices or preferences (explicit and/or implicit), the goal of this function is to infer possible user cooperative interactions or interests during the cooperative authoring work session.

2 Cooperative Editing

Cooperative editing field emerges from the works developed in other research domains such as distributed systems, broadcast networks, telecommunications, hypertext and databases [9][12][15].

The cooperative production of documents is the result of the coordination process between different authors to carry on the activities needed to the joint authoring production of documents. Then, co-authors will use cooperative editing tools to enhance their effectiveness, to enforce consistency, to control the validity of contributions, and finally to ensure the quality of the commonly produced work.

The application field of cooperative editing has a broad scope and includes technical documentation (aeronautic, car industry, office automation, encyclopedia), the scientific documentation (articles, reports, books), the medical documentation of patients in a hospital environment, etc.

In cooperative authoring applications the treatment of information can be supported following either the inter-document or the intra-document approaches. In the inter-document approach, the document being co-edited is organized as a set of objects potentially multimedia (annotations, notes, images, sounds, etc) which are linked using hypertext links. In the intra-document approach, a document is a single shared entity, and co-authors work in a private local editing environment, accessing the whole shared document.

Inter-document Approach (Hypertext)
In order to alleviate the lack of internal structuring, the shared document is arbitrarily decomposed and split into elementary and inter-linked parts (pages, notes, etc) and organized in a more complex structure. Moreover, using hypertext links to structure information, inevitably major management problems are generated, already well known in systems using a reference based upon addressing schema such as the Smalltalk system [7]. Among generated and relevant problems, we can note: link updating, garbage collection and dynamic detection of invalid links.

Despite the Web follows an hypertext organization where HTML or XML [1] pages are related using hyperlinks, our design is based on the utilization of the document structure in order to support and manage the cooperative contributions [3]. The hypertext link interest is not negated, and they are used for what they were designed: to link associated information (bibliographic references, side notes, more precise details, etc).

Intra-document Approach (Document Centered)
A document is a single entity shared by co-authors, each author works in a private local editing environment, access the shared document in its integrality, he can create elements according with the document structure. The cooperative editing system ensures the consistency of different contributions. The produced modifications are then synchronously or asynchronously viewed by other co-authors. The storage of documents can be either centralized in one site or replicated among several ones. In the case of replicated storage sites, the consistency and updating of different copies are automatically performed by the document management system. The cooperative editing system allows each individual user to perceive the commonly edited document as a whole, with the possibility of being notified of other author contributions.

Cooperative Editing of Structured Documents
A structured document editor uses the logical structure of a document class to guide the user actions during authoring phase of a document (instance of this class). Following this approach, a document is represented by its logical structure (*abstract tree*). This structure is mainly hierarchical, complemented by inter-elements links (e.g. reference to a section). The document partitioning based on its logical structure is one of the principles of our approach: the document is partitioned into shared units, ensuring the consistency of parallel and cooperative contributions [3].

In the Web environment, documents are structured (HTML), and can even be described and composed following generic document models (XML). In this

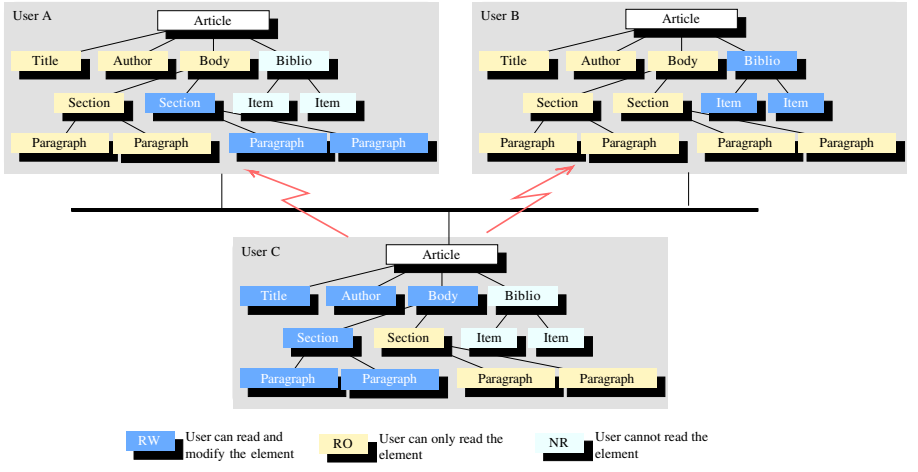


Fig. 1. Document Fragmentation and Editing Sharing

way, a great variety of documents can be designed (article, bibliography, chapter, medical document, technical report, etc) suited to various cases of applications. For instance, an article is organized as a set of elements: title, authors, sections, paragraphs, figures, diagnosis, etc, and logically integrated (see Fig. 1).

3 Web Cooperative Authoring

The current main developments in supporting Web cooperative authoring are being done by IETF (Internet Engineering Task Force): the WebDAV project [13][16], whose main goal is to extend the HyperText Transfer Protocol (HTTP) by adding new capabilities dedicated to cooperative production.

However, these extensions will not allow to support cooperative authoring of Web databases on the quite non-reliable Internet network. More precisely, an Internet connection between a client and the remote server processes establishes a connection chain that uses relaying sites, heterogeneous in terms of operating system natures, transmission rates, loading, nature of connection infrastructure, etc.

Related to the WebDAV proposition, we present the features of AllianceWeb: an authoring environment for cooperative production, consultation and management of Web documentations whose the cooperative editing function is based on the intra-document approach.

3.1 WebDAV

By extension of the HTTP protocol, special functions are defined for supporting concurrent and collaborative Web authoring of an entire document. Using these functions, a user can load a Web document, locally modify it and save it back on

the remote HTTP server (Fig. 2). To ensure the consistency of a concurrently edited document, WebDAV provides functions for handling locks (exclusive and shared) of remote resources. The concurrent production of a shared document follows these steps (Fig. 2):

1. The client-side authoring application sends to the server a request to initialize a lock on the document. If the locking operation is rejected then the resource authoring process is aborted.
2. If the locking operation succeeded, the document properties are returned.
3. The editing application gets a copy of the remote document.
4. The document copy is locally edited. The editing process can be supported using an interactive editing tool (Netscape, Amaya, etc) or a HTML/XML "tag" based tool (vi, emacs, etc).
5. Once the editing phase accomplished, the new version is transmitted and updated on the server.
6. Finally, the client application unlocks the remote resource.

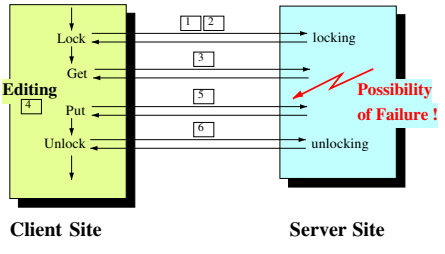


Fig. 2. WebDAV Cooperative Authoring Principle

WebDAV authoring is based on the well known "hybrid storage/processing" architectural principle (local treatment and remote data storage) that makes maintaining consistency of remotely accessed resources in an unreliable distributed environment difficult. This mainly comes from the potential unreliability of its three components (client site, remote storage site and communication network).

This model is commonly used in the Web environment to support browsing operations. In the case of authoring, browsing and modifications may be concurrently performed, so that we must ensure document consistency. Many problems may occur and several kinds of processing should be performed depending on failure sources (client, server and/or network) and on their associated characteristics (nature, time, locked resources).

The WebDAV solution is not suited for cooperative creation of large document collections on a wide area network as Internet (see Fig. 3) because of the high network failure frequency. Thus, this solution is only applicable to the case of a highly reliable network (e.g. a local area network in an industrial environment).

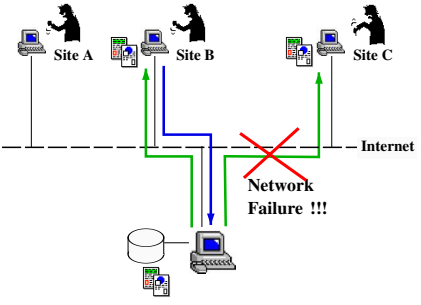


Fig. 3. Potential Problem with WebDAV

3.2 AllianceWeb: Cooperative Authoring of Web Documents

The goal of the AllianceWeb project is to determine the specific requirements and to design suited techniques to support future Web cooperative authoring environments which will allow users distributed on the Internet to produce the same documentation databases in a consistent way.

Each user has a global perception of the co-edited document. Functions allow him to notify his colleagues of his production. He is notified of other user contributions and more he is able to control the way by which they are integrated in his current version (*group awareness* function). The AllianceWeb project builds on the experience gained during the design and prototyping of the Alliance [3][4] cooperative editor on Internet.

Using elaborated functions to handle HTML or XML document structures, AllianceWeb allows manipulation of complex documentation. Thus, the document structure constitutes a base to apply efficient treatments on its content. In addition, it constitutes a suited support for the expression and management of the user cooperation. In this way, the structure of documents is used to define the units of sharing called **fragments** on which authors may act based on their **editing roles**.

In AllianceWeb, cooperative authoring is based on the three following principles:

1. **Document partitioning** which consists of splitting a document into several **fragments** following its structure (from this principle, sharing granularity becomes a essential feature),
2. **Assignment of editing roles** (manager, writer, reader or null role) for co-authors of the document,
3. **Management of group awareness** that controls document development allowing users to notify other users and to be aware of their contributions.

The user who cooperates to the production of a shared document, can simultaneously act with several roles on the different document parts (see Fig. 1). Thus, a user who acts as “writer” on a fragment can modify it, and in the same time he only will be able to consult fragments on which he acts as “reader”.

The "null" role is defined to prevent consultations of a fragment that must be considered as confidential. The last role, called "manager", allows users having it to assign or change roles of other users, and more to dynamically split or merge fragments.

A fragment may contain all elements of the document logical structure, for example a chapter, a section, a paragraph, a list of elements or simply a word. Thus, the document fragmentation is dynamic, and the sharing granularity is variable.

4 Adaptive and Deductive Group Awareness Function

The group awareness function is a design key feature of a cooperative application. From the already defined and basic group awareness function elaborated in the Alliance experiment [2], AllianceWeb allows each user:

1. to diffuse his contribution to other users,
2. to control the integration of other contributions in his working environment.

From this base, we want to define more sophisticated principles and more powerfull functions to allow users to better perceive, view, understand and interact in the cooperative authoring process.

AllianceWeb provides an asynchronous notification mechanism (delay between the action and its perception) that allows user to be notified of the editing work produced by his colleagues. Currently, the group awareness function allows each user to configure the notification level (attentive reading, automatic updating or unattended reading) for each fragment he is consulting.

First, considering the object space manipulated by co-authors (documents, fragments, associated resources, communication objects such as dialog windows, "chat" tools, multimedia electronic conference tools, etc, and author objects themselves), and second the set of actions applied on the object space (tool execution, selection and/or modification of a document part, production or consultation of an annotation, etc), the goal is to define a flexible, deductive and adaptive cooperative group awareness environment.

- **Deductive** - The system is able to analyze the cooperative working actions of each user, and analyzing the action sequences, their number and frequency, discovering some localities, etc, it is able to determine user objectives.
- **Adaptive** - Taking into account each user objectives, and in respect with their individual preferences (eventually antagonists), it guides and dynamically adapts the user environments. By this, the goal is to define an "intelligent" part able to detect possible cooperative points of interest in order to facilitate user cooperation.

In order to avoid users have to manages their cooperation environments and to provide him more efficient and productive cooperation work, our system uses an inference engine (see section 4.1). This engine automatically manage the group awareness environment (e.g. What are the suited tools to highlight in the user

environment to make him more efficient in the cooperation process? On what user activities is it judicious to concentrate interaction or group awareness tools? etc).

Thus, using a dedicated and cooperative interaction interface, user is able to easily export his contribution, and he can integrate other user contributions in a so comfortable and easy way. More, he takes benefits from the automatic updating of his cooperation interface. User is discharged of all cooperation support management duties and his actions are then only directed to the cooperation goals.

4.1 A Group Awareness Inference Engine

The goal of the Group Awareness Inference Engine is to determine the manipulated entities and the executed cooperation actions, to be able to generate and/or modify the presentation of these informations on the user screen. It is important to note this presentation is constrained by user preferences applied using presentation filters.

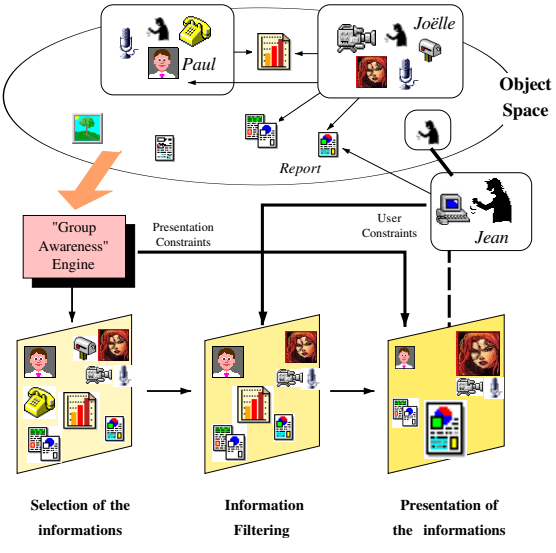


Fig. 4. Group Awareness Engine Principle

Periodically the inference engine re-evaluates and updates the set of manipulated objects (users, private and/or public objects, shared objects with some other users, date of actions, etc). This periodic re-evaluation is performed in parallel with the actions and treatments applied by the user (e.g. *Jean* in Fig. 4). This re-evaluation is completed both by statistic measures of the manipulated object space and by the analyze of the concurrent interactions (**concerned users, manipulated objects and action frequency**).

The group awareness inference engine is guided by **user preferences** (figure organization, formats, presentation, tool interfaces, etc) and **user role** on the shared fragments.

For instance, let us consider the cooperative environment of user Jean, who mainly acts as a consultant (reviewing, annotations, revisions) on the major part of the shared document. The inference engine has a rule that evaluates his environment and automatic updates it with suitable tools to better cooperate:

```
If author (document) = x
if role (x) = ‘‘reader’’
if frequency (‘‘reader’’) > 0.5
if annotationTool (x) ≠ ‘‘installed’’
Then
  InstallInEnvironment (x) ← annotationTool
EndIf
```

The base of rules uses first order logic language to write rules as it is described in [8][10] . Concerning our example, the inference engine triggers specific cooperation actions (annotation tool) in the environment of user *Jean* (“reader”). This is the way to close him to the cooperative group production.

After that, the set of selected objects is filtered in respect with indications and constraints specified by cooperating users:

```
If author (document) = x
if comunicationTool (x) = ‘‘refused’’
Then
  comunicationTool (x) ← TurnOff
EndIf
```

Thus, assuming *Jean* indicates he is not interested to synchronously communicate (electronic audio/video dialog) with user *Joëlle*. Moreover, user *Joëlle* has not authorized annotations of the fragment she is currently editing, since she prefers to directly interact with the persons interested by her current contribution. The rule which notify all “reader” users of the current fragment restrictions imposed by the user *Joëlle* is the following:

```
If author (document) = x
if role (x) = ‘‘writer’’
if annotation (x) = ‘‘refused’’
if communication (x) = ‘‘open’’
if author (document) = y
if role (y) = ‘‘reader’’
Then
  Announce (y) ← OpenCommunication with x
EndIf
```

Moreover, if user *Jean* is strongly interested in *Joëlle's* contribution, we have a conflict between the tool filtering process expressed by Jean, his strong interest

for *Joëlle's* work, and the *Joëlle's* choice to directly interact with people. The group awareness inference engine evaluates the preferences of these two users, and if a conflict is presented, it confronts them, and it reachs a way (decision) to modify the two cooperation environments.

Thus, Jean's restrictions are expressed by the rule:

```
If author (document) = y
if comunicationType (y) ≠ ''synchronous''
if author (document) = x
Then
  Announce (x) ← CloseCommunication with y
EndIf
```

The system announces to all users the availability degree of everyone in the global cooperation process. Then, each user may decide to change or not his interest or his restrictions. That is the way to solve the current conflict.

The following rule expresses the fact a drawing document is not interesting for the user.

```
If author (document) = x
if documentType (document) = y
if interest (x) ≠ y
Then
  editingTools (x) ← TurnOff
EndIf
```

Jean cooperatively works with *Joëlle* on the document *Report*. The inference engine maximizes the tools that allow to establish and to support an intensive and efficient cooperative work between *Joëlle* and *Jean*: cooperative authoring, multimedia communications, annotation and negotiation tools, etc. In the same time, this engine minimizes the perception of user *Paul* in the *Jean*'s environment.

Thus, the presentation of each user environment is re-evaluated, and some presentation constraints are automatically generated by the inference engine to reflect the importance and/or the punctuality of the resource interactions. In this way, the inference engine may take decisions in opposition with some user preferences.

We can also foresee designing strategies which will allow to define some expert levels for users, and thus to permit more or less experimented users to take benefits of expert knowledge.

For example, a user who usually produces a lot of figures will be characterized as "illustrator". When another co-author (not classified as "illustrator") who wants to produce a drawing, and who makes a lot of hesitations, the system will propose him to consult the figure base of the "illustrator" specialist. In the same way, we can imagine his interface will be updated to integrate some synchronous cooperation and/or communication tools to contact the specialist. Possibly, this interface will be completed to integrate some on line help tools.

```
If InsertedNature (document) = ‘‘figure’’
if author (document) = x
if role (x) = ‘‘writer’’
if classification (x) ≠ ‘‘illustrator’’
if author (document) = y
if classification (y) = ‘‘illustrator’’
Then
  Announce (x) ← DrawBaseConsultation (y)
  Announce (x) ← OpenCommunication with y
EndIf
```

4.2 The Object and Action Spaces

In order to define a deductive and adaptive inference engine to support the group awareness function, it is required to identify the manipulated objects, and the actions performed on these objects. Of course, authors are also represented in the system as special objects. Thus, among all objects managed by this system, we may identify:

- **The Documents**
  The documents are considered as global entities but even as object compositions (fragments, images, video, audio, various metadata, etc) to support information partitioning and sharing.
- **Some Special Cooperative Authoring Objects**
  In addition to the AllianceWeb cooperative editor, some other cooperative tools may be present in the user environment: an electronic white board, a multimedia teleconferencing tool, a “chat” tool, etc. All these tools may be partially or fully integrated in the AllianceWeb environment to enhance group cooperation.
- **Objects for Group Awareness Support**
  Some specific group awareness tools are also defined in the user environment to build or update his interface. These tools allow to consult the state variables (availability and permission) of other user environments:
  - Can I open a video / audio connection with user “X” ?
  - Can I look at the draft of my colleague “Y” ?
  - Is my colleague “Z” free to open a synchronous (“highly coupled”) editing session of this paragraph?

  These informations show the co-authors cooperation states (current work, availability, points of interest, set of available interaction tools, etc), and more precisely they show the authorized user interaction possibilities.
- **Actions Applied on Documents**
  The AllianceWeb editor allows to cooperatively author HTML and XML documents, and to manage their associated resources in a structured and consistent way.
  Among the observable actions, we can identify:
  - document creation, renaming or removing,
  - document editing or consultation,

  - adding, modification or removing of a document resource,
  - document fragmentation,
  - diffusion of a new fragment version (output operation) or fragment updating (input operation),
  - modification of the user editing session focus,
  - adding, modification or consultation of an annotation.
- **Document Editing Actions**
  The editing kernel on which AllianceWeb is built allows to catch and control all editing events (actions) explicitly performed by user or indirectly triggered as consequence of other actions (double clicks on a bibliographix reference).

The observable action list is not restricted, and many other events explicitly or implicitly generated by editing, resource managing, cooperative activities management, etc can be used as input information for the group awareness inference engine.

AllianceWeb architecture
Taking into account the system support needs of the AllianceWeb application (author and document naming, document storage, accesses to (remote) documents, group awareness management, information protection and confidentiality, etc), we use the World Wide Web technology as the basic communication infrastructure to support access to remote information, and to manage the distributed or replicated information:

1. The HTTP protocol which allows to establish stateless connections between a client and the remote Web server processes,
2. the Web resource naming system based on the URL notion ("*Uniform Resource Locator*") on which we define the AllianceWeb user, document and resource naming system,
3. and the possibility to extend the Web server functionalities making calls of pre-installed CGI scripts. A CGI script is a standard program that may be called to perform specific actions on Web server resources (HTML or XML documents), or on other kind of resources (files, processes, databases, or any other kind of specifically structured resources).

Because on the unreliability of the Internet support and services, we separate the AllianceWeb editing function and the communication service (see Fig. 5). The goal is to make AllianceWeb failure tolerant (server or network failures). Thus, this choice allows to define AllianceWeb as a distributed cooperative editing environment, insensible to Internet network delays: this property allows to apply AllianceWeb techniques in temporarily disconnected environment and more to be used as a support for nomadic editing work.

Thus, each running instance of the AllianceWeb application is composed of two separated active parts (processes): the *browser/editor* and the *assistant*. During the user work session, these two components cooperate closely:

- *The browser/editor* goal is to manage user interactions: management of the document displaying and interpretation of user editing commands. All editing commands uniquely generate modification actions of the local document

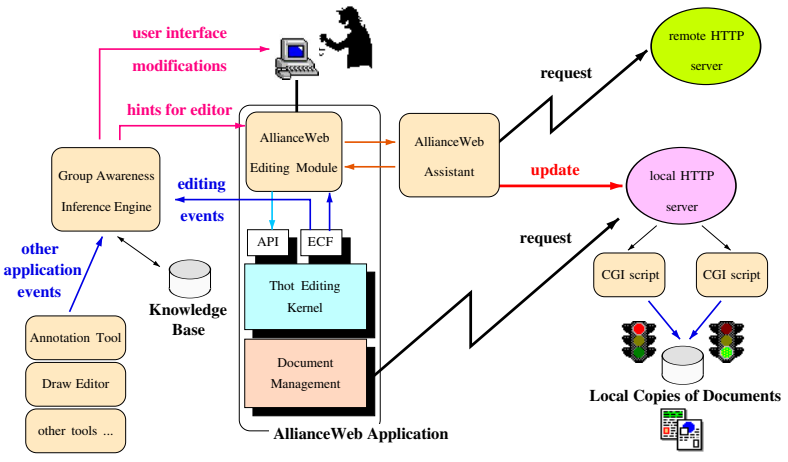


Fig. 5. AllianceWeb and the Group Awareness Inference Engine

copies. Eventually, they generate some asynchronous remote requests that are delegated to the assistant. These requests are non blocking operation such as the evaluation of a new editing opportunity: for example, "Can I obtain a higher role on this fragment?"

– *The AllianceWeb assistant* constitutes the AllianceWeb interface with the remote AllianceWeb servers. It takes in charge all communication problems (site and network failures), and thus periodically it tries to recontact all temporarily unreachable remote servers. This component works in standalone and autonomous mode.

It is important to note that these two components concurrently make access to the same local document base (see Fig. 5): the local browser/editor loads local document in main memory and applies user modifications. The assistant receives events to update the local copy of the document, regularly consults other remote document bases, and frequently integrate new fragment versions in the local document base.

Editing Action Handling

The AllianceWeb browser/editor is built on top of the Thot editing kernel [11] uses a dedicated document management module that offers specific functions: naming, document and resource accessing, resource sharing and updating.

The editing kernel provides highly structured editing functions to handle HTML or XML documents in a powerful and fine grained way. These functions are accessible using an API (*"Application Programming Interface"*). In addition, using the ECF (*"External Call Facility"*) functionality (see Fig. 5), all events generated by the editing kernel are passed to the application module and to the group awareness inference engine. Each event can be controlled by the application that can decide to complement, to replace or to ignore the associated default treatment.

5 Conclusion

After presenting the field of cooperative edition and analyzed the limitations of current work being done in Web-based environments, we have defined the principles (documents fragmentation, role attribution and group awareness support) of the AllianceWeb environment.

AllianceWeb allows several authors distributed over the Internet to produce shared Web documents in a concerted and consistent way; for this we have created a seamless production/consultation space where co-authors cooperatively produce shared documentation. At the same time, all users doing Web navigation can consult the documents whose versions are automatically and regularly updated.

The group awareness function, which is an essential component of all cooperative applications, allows each user to be aware of the production of others and allows them not only to adjust their contribution, but also to influence or to guide other user contributions. The interaction interface, used by each user to perceive his work and other user works, becomes very complex and difficult to manage: the number of objects (users, tools, documents, shared objects, etc) that are parts of his environment is too large, and more over time the importance and interests of user participating in the cooperative work regularly evolve.

We have presented the complexity of the group awareness function, and showed the interest of the AI field to implement it. We have presented the first elements and principles of a group awareness inference engine that will allow the automatic adaptation of each user's interaction environment, taking into account their preferences and their actions in the shared objects space.

It is important to acknowledge that the groupware field, after its initial interest in supporting cooperative work (organization and management of collective work), has naturally moved towards distributed environments, such as the Web. This new dimension enables remotely located users to cooperate and produce information in a concerted and consistent way. This way, all group awareness support problems, already present in centralized systems, have been considerably increased.

The principles and techniques deriving from Artificial Intelligence domain appear very useful to design and implement effective and powerful cooperative environments. We have proposed the first guidelines of action to support deductive and adaptive group awareness for producing documents in a Web environment. We see this as a new field of multidisciplinary research that shows great promise.

References

1. N. Bradley, "The XML Companion", First edition, Addison-Wesley Pub. Co., September 1998.
2. D. Decouchant, V. Quint et M. Romero Salcedo, "Structured Cooperative Editing and Group Awareness", *In Proc. HCI International '95, 6th International Conference on Human-Computer Interaction,* Yokohama, Japan, 9-14 July 1995.

3. D. Decouchant, V. Quint et M. Romero Salcedo, "Structured and Distributed Cooperative Editing in a Large Scale Network", *Groupware and Authoring (Chapter 13),* R. Rada, ed., pp. 265-295, Academic Press, London, Great Britain, May 1996.
4. D. Decouchant et M. Romero Salcedo, "Requirements and Design Issues for a Cooperative Authoring Application", *Third International Workshop on Groupware (CRIWG'97),* pp. 111-118, Madrid, Spain, 1-3 October 1997.
5. D. Decouchant, A. M. Martínez et E. Martínez, "Documents for Web Cooperative Authoring", *In Proc. CRIWG'99, 5th CYTED-RITOS International Workshop on Groupware,* IEEE Computer Society, Cancun, Mexico, 15-18 September 1999.
6. P. Dourish and S. Bly, "Portholes: Supporting Awareness in a Distributed Work Group", *Proceedings of CHI'92 on Human Factors in Computing Systems,* P. Bauersfeld, J. Bennett and G. Lynch, ed., pp. 541-547, Monterey (California), May 1992.
7. A. Golberg et D. Robson, "Smalltalk-80: the Language and its Implementation", Series in Computer Science, Addison Wesley Publishing Company, 1983.
8. E. A. Feigenbaum and P. McCorduck, "The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World", Addison-Wesley, 1983.
9. R. S. Fish, R. E. Kraut, M. D. P. Leland et M. Cohen, "Quilt: A Collaborative Tool for Cooperative Writing", *Proceedings of Conference on Office Information Systems,* pp. 30-37, ACM Press, 1988.
10. J. L. Laurière., "Intelligence Artificielle, résolution de problèmes par l'homme et par la machine", Eyrolles Press, Paris, France, 1986 (in french).
11. V. Quint and I. Vatton, "Making Structured Documents Active", *Electronic Publishing - Origination, Dissemination and Design,* vol. 7, num. 1, pp. 55-74, March 1994.
12. N. Streitz, J. Haake, J. Hannemann, A. Lemke, W. Schler, H. Schütt and M. Thüring, "SEPIA: A Cooperative Hypermedia Authoring Environment", *Proceedings of the European Conference on Hypertext and Hypermedia ECHT'92,* D. Lucarella, J. Nanard, M. Nanard, P. Paolini, ed., pp. 11-22, ACM Press, November 1992.
13. R. Khare, J. Whitehead, A. Rifkin, "Web Automation & Collaboration", California Software Symposium, University of California, Irvine, October 1998.
14. J. C. Lauwers and K. A. Lantz, "Collaboration Awareness in Support of Collaboration Transparency: Requirements for the Next Generation of Shared Window Systems", *Proceedings of CHI'90 on Human Factors in Computing Systems,* J. Carrasco Chew and J. Whiteside, ed., pp. 303-311, Seattle (Washington), April 1990.
15. C. M. Neuwirth, D. S. Kaufer, R. Chandhok et J. H. Morris, "Issues in the Design of Computer Support for Co-authoring and Commenting", *Proceedings of the Conference on Computer Supported Cooperative Work,* pp. 183-195, ACM Press, October 1990.
16. E. J. Whitehead Jr., M. Wiggins, "WEBDAV: IETF Standard for Collaborative Authoring on the Web", *IEEE Internet Computing,* pp. 34-40, September - October 1998.

Definition of a General Conceptualization Method for the Expert Knowledge

Almudena Sierra-Alonso

Universidad Carlos III. Escuela Politécnica Superior.
Avda. de la Universidad nº 30, 28911 Leganés, Madrid, Spain
asierra@ia.uc3m.es

Abstract. The community of knowledge engineers usually accepts the importance of the analysis process of the expert knowledge. This analysis, also known as conceptualization, is carried out at the knowledge level. The objective of this analysis is to obtain a conceptual model, from the knowledge acquired from the expert. The conceptual model collects the relevant parts of the reality for the problem solution, forgetting those elements that are not meaningful for the system. Furthermore, it permits that the knowledge engineer (KE) to understand the problem domain and to detect knowledge gaps or wrong interpretations. There are several proposals to accomplish this analysis at knowledge level, but they do not offer a detailed process that helps in the elaboration of the conceptual model. This work proposes a method to guide in the elaboration of the conceptual model. KEY WORDS: Knowledge based System, Conceptualization, Conceptual Models.

1 Introduction

In the development of a Knowledge Based System (KBS), the analysis and modeling knowledge process, is very important. This importance is due to the objective of this process is to understand and to model the knowledge that the system will use to reason. The quality on the KBS depends on the quality of the knowledge model. This conceptual model must allow:

- To make tangible the real world concepts and relationships.
- To make explicit the implicit or tacit knowledge that the expert handles.
- To emphasize the relevant parts of the reality for the problem solution, abstracting those elements that they are not meaningful for the system.
- To know and to understand the problem and its environment.
- To detect knowledge gaps.
- To locate mistakes or wrong interpretations before advancing in the construction process of the system.

Once a problem is recognized in a domain of the real world, the KE identifies "raw" knowledge pieces that the expert uses to solve the problem. It is necessary to analyze, to combine and to structure these knowledge pieces so that they form a model of the expert knowledge. This conceptual model reflects and describes the problem

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 458—469, 2000.

and the way to solve that problem. Afterwards, the conceptual model is transformed to express the knowledge through representation formalisms. Finally, it is necessary to write the model in terms of a programming language, to obtain the implementation of the model.

There are tools and techniques to guide those transformations. But the methods that guide in the transformation of the independent knowledge pieces into the conceptual model are not completely useful. As we will see in next section, the majority of existing methods define structures and patterns to model the knowledge, but they do not give a detailed guidance to elaborate the model using those predefined structures.

The objective of this work is to define a method to elaborate conceptual models in the KBS area, to save the existing gap in the conceptualization stage. This method will be of general character, applicable to several types of tasks and will help to transform the knowledge, to obtain the conceptual model. The method consists of a representation language and the process to build the model. The language will include the basic concepts to represent the elements of real world. Both the language and the process will be located at knowledge level and, therefore they will be independent of the implementation, as well as of the type of task. Another objective of this work is to make a special point of the revisions and tests to prove the model validity.

2 Bibliographical Review

The first projects to model the expertise were based on production rules and frameworks [8]. These representations belong at symbolic level and they are too nearby to the machine. This representation form was a consequence of an outdated view of the knowledge engineering: to build a KBS was a process of knowledge transfer from the expert to the machine [19]. But already Newell, had indicated the need of representing the knowledge at a removed level from the machine, that he called knowledge level. In this greater abstraction level, the knowledge representation depends on the form in which that knowledge is going to use [14]. With the representations at symbolic level this dependency did not take into account [6]. Then, new proposals to represent and use the knowledge emerge. Now, the knowledge engineering is seen as a modeling process instead of a transfer process [17, 20]. The existing proposals to represent the knowledge can be classified into two groups:

- Works that propose to model specific methods for each type of task[1], according to the idea that the knowledge representation depends on its use and the inference strategy to be applied to the knowledge (interaction problem [3]). These methods are known as problem resolution methods (PSM).
- Works that propose a general method to elaborate the conceptual model. They use the PSMs and domain ontologies. But also they define a method to build the conceptual model if no PSM or ontology is available. These works are usually methodologies that encompass the whole KBS construction process.

[1] "Type of task" or "type of problem" are for example: classification, diagnostic, interpretation, planning, assignment, design, monitoring, etc.

2.1 Specific Methods for Conceptual Modeling

The works that propose specific methods are based on the analysis of built systems. This analysis abstracts forms of reasoning depending on the type of task. Clancey was the first who observed a similar behavior in several developed expert systems and identified a PSM that called heuristic classification [9]. But the concrete domain knowledge is represented using production rules, frameworks or LISP [20]. Since Clancey's work, other approaches are developed PSMs: the role-limiting methods [12], the generic tasks [4, 5] and the tasks structures [19]. These proposals share the following features:

- They analyze KBS in abstraction levels far from the implementation level [8].
- They identify strategies narrowly associated with the task and the knowledge types that the resolution strategies of those tasks require [8].
- For each type of task, both the resolution method and required domain knowledge are determined [7, 20].
- They try to guide the knowledge acquisition through the knowledge types required by each resolution method [7].

But these approximations present certain problems:

- There is no an agreement on which are the basic types of expert tasks and how they are solved. In diagnostic and classification tasks, there is some agreement. But for other tasks (for example interpretation), it is not very clear what tasks are generic and the decomposition level that has to be reached [19].
- The structures for all types of tasks have not been established. Therefore, these approximations only serve to solve some types of problems.
- They are centered mainly on the method to solve the problem (indicated in Fig. 1 as Specific Methods). Though there is an underlying knowledge model for each one of those methods, the representation language is not clearly defined. Even some of those models is represented at symbolic level, for example using rules in the heuristic classification, and not at knowledge level [9]. This carries to a confusion: is it wished to represent conceptual models or formal models?
- They lack a detailed guidance that helps KE to build the conceptual model (indicated in Fig. 2 as Conceptual model elaboration Method).

2.2 General Methods for Conceptual Models Construction

As consequence of the problems presented above, the need of new methods emerges. These methods have to provide a language to represent models at knowledge level, to establish the steps to elaborate the conceptual models and to give a solution if there no is a PSM for the problem to solve.

These methods to build conceptual models are a part of the development methodologies that encompass the complete construction process of KBS, though in this work only the conceptualization phase will be considered. In this line are the methodologies KADS [17], MIKE [2, 20], PROTEGÉ-II [1, 15], KSM [13] and IDEAL [10]. These proposals present the following characteristics:

- They are located at knowledge level to model the problem.
- They propose general modeling languages to describe at knowledge level the independent conceptual model of the type of task. KADS, PROTEGÉ-II and MIKE languages are defined formally. The language proposed in IDEAL is not complete. KSM does not propose any own language, but permits to use anyone of the languages defined in the others methodologies.
- They permit the incorporation of PSM and ontologies**.**

But the analyzed works lack a detailed and formalized process that guide to the KE in the construction of the conceptual models using the proposed language. So that the difficulty to obtain a conceptual model from raw knowledge pieces persists. These methods establish what models are to be elaborated, but not how to elaborate them. Though not all in the same degree: for example KADS, that is defined as a declarative methodology [17, chapter 7], lacks that process. However, IDEAL proposes some general heuristics to build the model [10].

3 Principles of the Conceptualization Method

To overcome the problem described above we present a general conceptualization method (GCM) that provides a detailed guide to construct and validate the conceptual model. Before describing this method it is necessary to reflect on three problems:

- The nature of the knowledge that it is necessary to represent in the model.
- The activities the KE accomplishes when he/she elaborates a conceptual model.
- The close relationship between acquisition and conceptualization phases.

3.1 Knowledge Levels

The acquired knowledge does not always have the same degree of detail, neither the same function. According to the function that accomplish the knowledge in the domain, three knowledge levels are distinguished, as IDEAL proposes [11]:

- Factual knowledge. It specifies with what the reasoning is made. Properties, concepts, relationships, etc are defined. This knowledge defines the domain structure and other knowledge levels use it for reasoning.
- Tactical knowledge. This is the knowledge about the particular reasoning to solve the problem. It specifies how the reasoning is made.
- Strategic knowledge. This is the knowledge about the strategies to solve the problem. It specifies what steps should be followed to reasoning and when they have to be accomplished. It is necessary to define: the steps to fulfill the task, its sequence and performance conditions and what information is transmitted from a step to others. Basically it forms the process and control model.

These three knowledge levels are related between them. The strategies control the reasoning. The knowledge about how to reason needs concrete elements to reason. These elements are the factual knowledge.

3.2 Activities of the Knowledge Engineer

Once the knowledge is acquired, it should be structured. Two activities are carried out: one of identification (analysis) and other of organization (synthesis). Finally, it is necessary to prove that the model is correct. This final test must join the three types of knowledge in order to accomplish a holistic test [11]. Therefore, we can say that the conceptualization is carried out in three parts:

- Analysis activity. In the analysis process of each acquisition meeting the knowledge is identified by differentiating the different levels.
- Synthesis work. In the synthesis process the knowledge is organized using the proposed representation language to find the problem solution.
- Holistic test. A conceptual model function is to detect knowledge gaps, wrong interpretations and mistakes. The methods presented before do not make a special point of this objective. This work insists on this question. The objective of this activity is to accomplish a validation of the knowledge reflected in the conceptual model in order to prove if that knowledge represents correctly the wished behavior. This validation is carried out for each knowledge level, since this test incorporates the three levels of knowledge in a single representation: the knowledge map and the process model.

These processes, mainly analysis and synthesis, are not sequential but simultaneous a line to indicate the end of the analysis and the beginning of the synthesis cannot be established. In fig. 2 the coincidence between analysis and synthesis can be observed.

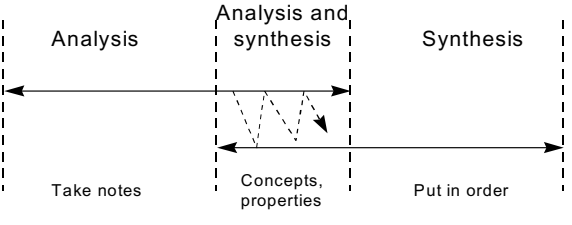


Fig. 1. Overlapping between analysis and synthesis.

When the conceptualization begins notes are taken, terms are underlined, ideas are selected, etc; that is, the analysis is realized. After some time but simultaneously with the previous activities, relationships between those terms are found out, it is decided whether these terms are concepts or properties and classifications are established. That is, the analysis and synthesis are mixed. Finally the notes and the selected knowledge are organized. That is, synthesis is realized.

3.3 Relationship between Acquisition and Conceptualization

The conceptualization stage is strongly bound with the acquisition stage. In fact "acquisition" is found in the literature to name the process that encompass from the acquisition to the conceptual model elaboration [17]. Fig. 3 shows the elicitation cycle [10]. The step "Analysis of acquisition meeting" is the conceptualization stage. In this

step the knowledge is analyzed. As consequence of the analysis new aspects emerge that have to be treated in depth. Then, other acquisition meeting is necessary. This relationship is clear to experimented KE, but inexperienced engineers have difficulties to notice that the analysis of a meeting and conceptualization are the same activity.

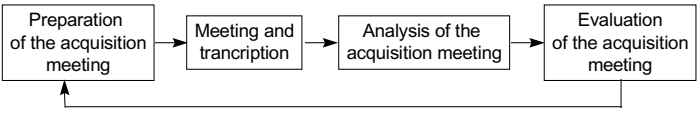


Fig. 2. Elicitation cycle.

4 Conceptualization Method

The proposed method, GCM, is independent of the type of task, thought it permits to incorporate the PSM and domain ontologies. In this paper we are centered in the method to elaborate a new model when no PSM or ontology is suitable for the problem to solve. To define the method is necessary to establish:

- The representation language to describe the conceptual model.
- The steps that guide the construction of the model using that language.

4.1 Representation Language

The definition of a language is not the principal objective of this work because it is already well defined in the literature. However, it is necessary to establish which representation forms we are going to use. In this work the representation language has been defined using languages and formalisms in existing software engineering, in databases and in artificial intelligence.

- Factual knowledge. The elements to be represented at this level are: concepts, properties, property values, relationships, restrictions on properties or on relationships, and instances. The language used to represent these elements is the customary in the objects models [16].
- Tactical knowledge. At this level calculations and inferences have to be represented. The formalisms selected to represent the forms of reasoning are:
  - For the inferences: rules, decision trees, decision tables or inference nets.
  - For the calculations: mathematic expressions or algorithms.
- Strategic knowledge. The strategic knowledge are the steps the expert follows to solve the task. The identification of those steps includes several aspects:
  - The task decomposition. If the decomposition is hierarchical a decomposition tree can be used. If the steps to fulfill the task have no hierarchical relationship, a net of relationships is used. For each step that is not decomposed, it is necessary to describe its function and how it is carried out.
  - Steps sequence and conditions for task fulfillment. If the sequence is determinist, flowcharts are used to represent it. If it is non-determinist rules and metarules or Petri nets are selected.

4.2 Conceptualization Process: Steps in the Analysis of the Acquisition Meetings

The conceptual model takes shape step by step in each acquisition meeting. Therefore, the analysis of each meeting is an important part of the conceptualization.

Below, the activities that form the conceptualization stage of an acquisition meeting are presented. These activities are carried out simultaneously. The actual first activity depends on the following factors: (1) experience of the KE; (2) the engineer's experience in the domain. If the domain is well known, meetings devoted to the terminology are reduced; (3) the moment of the analysis. In the first acquisition meetings factual knowledge is obtained. In the intermediate meetings tactical and strategic knowledge is acquired and the last ones are usually dedicated to debug and refine the conceptual model.

The conceptualization of an acquisition meeting is realized in the following steps: (1) glossary; (2) to identify knowledge at different levels: factual, tactical and strategic; (3) revisions: several tests have to be made to determine inconsistencies, knowledge gaps, etc.; (4) knowledge map; and (5) process model.

Fig. 5 shows how the conceptualization stage fits into the elicitation cycle. We can see that the steps 1, 2 and 3 cover the analysis and synthesis activities. The holistic test is realized with the knowledge map (step 4) and the process models (step 5).

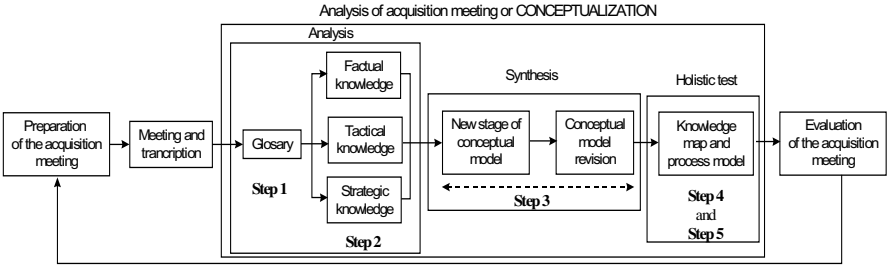


Fig. 3. Steps in conceptualization stage.

Step 1. Glossary

To extract the terms for the glossary by underlining the words and expressions that refers to elements that seem important in the domain.

Step 2. Knowledge Identification at Different Levels

Step 2.1. Identify Factual Knowledge

The activities to accomplish this step are the following: (1) to identify types of concepts; (2) to identify properties and values; (3) to identify restrictions; (4) to identify relationships between concepts and (5) to draw the relationships graph.

The activities of this step are shown in fig. 6. During the synthesis process the documents that describe the static model are elaborated: the concepts dictionary, the table concept/property/value, and the relationships graph.

The analysis process starts with the identification of these elements, but a rigorous order does not exist. For example, in an expert's sentence there can be a concept and properties of that concept, or two concepts and a relationship between them. In this

case, concept and properties or concepts and relationship would be identified at the same time. That is, the activities of this step do not have to be made sequentially. They can be made simultaneously.

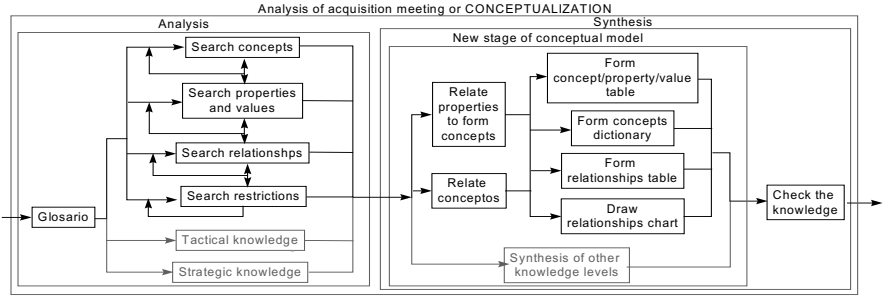


Fig. 4. Detail of the factual knowledge identification process.

Step 2.2. Identify Tactical Knowledge

The inference knowledge usually is rule-like. In an acquisition meeting there can be sentences that follow determined patterns. For example: "if certain situation is given, this is made" [19]. When the number of rules grows, some of those inferences could be structured in a tree or a decision table. This one facilitates the use of that knowledge and the reviews, to detect knowledge gaps, inconsistencies or any mistake.

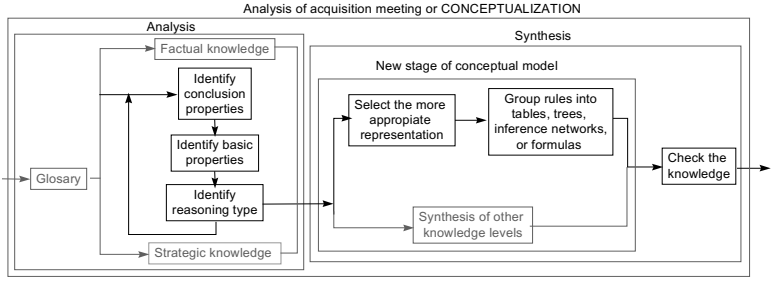


Fig. 5. Detail of the tactical knowledge identification process

To model the tactical knowledge follows these steps:

- To seek inferences. To determine the representation of those inferences (decision tree, decision table or inference net) follow the next steps: (1) to identify conclusion properties and to select the rules that concludes values for the same property; (2) to identify which properties are used to deduce those conclusion properties; (3) IF the basic properties are a set of properties, always the same, that are repeated in the left part of those all rules THEN define a tree or a table; (4) IF the basic properties are different and a repeated set of them can not be established in a group of rules THEN the rules form an inference net.
- To seek procedures. If a property is calculated mathematically or in a determinist way (algorithm) use a formula.
- Assign that tactical knowledge to a strategic step. The splitting in steps can change. In this case, check and allocate again the defined tactical knowledge.

Step 2.3. Strategic Knowledge Identification

In this step the task steps and substeps and their sequence are identified. Furthermore, each step is defined establishing: inputs, outputs, fulfillment conditions and the reasoning to achieve that step. The activities to follow are: (1) to identify the steps in which the task can be decomposed. Establish whether there is hierarchic dependency between the steps. This will determine the use of a hierarchic decomposition tree or a relationship between steps; (2) to identify the sequence of these steps. Whether it is determinist and the knowledge that directs the non-determinist performance; (3) to identify the conditions in which each step would be carried out; (4) for each step, to study a new division in substeps. To repeat steps 2, 3 and 4; (5) to define the substeps: step purpose, fulfillment conditions (other step must be fulfill before or restrictions over property values), input data (how the step starts, additional required knowledge), the reasoning carried out in that step (inferences or calculations to accomplish that step) and establish the use for the outputs.

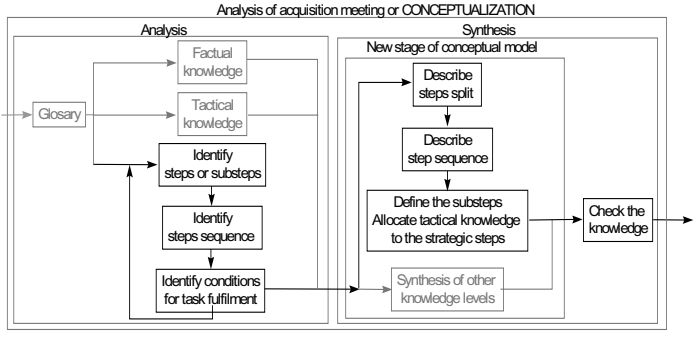


Fig. 6. Detail of the strategic knowledge identification process.

Step 2.4. Reviews

In the conceptual model can have mistakes or knowledge gaps that must be detected. We see that two of the conceptualization objectives were: to detect knowledge gaps and to locate mistakes before advancing in the system construction. Therefore, when an elicitation cycle finishes, it is indispensable to accomplish a review activity. This review is centered in two types of mistakes: mistakes in the model definition and mistakes, inconsistencies or gaps in the knowledge that describes the model.

The GCM method details the concrete actions that it is necessary to accomplish over each part of the conceptual model. These actions are completely defined in [18]. Because of space constraints only the targets over which action have to be carried out are mentioned:

- Factual knowledge. Actions are defined over the concepts dictionary, relationships table, concept/property/value table and relationships chart.
- Tactical knowledge. Actions are defined over decision tables, rules, decision trees, formula definition and inferences net.
- Strategic knowledge. Actions are defined to review the decomposition task, the steps sequence and the step definition.

The problems that appear during these revisions in a knowledge level can lead to detect problems in other level. The detected problems have to be taken into account in the following acquisition meetings. Furthermore, an integrated revision of all knowledge levels is necessary. This revision is carried out in the next paragraphs.

Step 3. Knowledge Map

The knowledge map (KM) is a mental diagram that allows complex ideas to be logically set out easily and quickly. A KM is a graphic representation of the links between properties and the process of deducing values of those properties [11].

In the knowledge map the expert reasoning is represented integrating static knowledge (the attributes) with dynamic knowledge (the connections between properties represent the inferences). Therefore, the KM is the holistic test of the conceptual model because it represents the KBS performance.

To build the knowledge map six steps are defined. This steps are described in detail in [11]. The steps are: (1) to identify the system objective; (2) to design the goal property for that objective; (3) to add the properties that are used to deduce the goal property; (4) to extend the KM adding the properties used to deduce the properties added in the step 2; (5) to repeat step 4; and (6) to check the KM.

Step 4. Process Model

The process model is another test and validation tool. The objective is to prove that for each strategic step, the necessary concepts and relationships have been defined. So this tool integrates the strategic and factual (concepts and relationships) knowledge.

As the knowledge map, it is a holistic test tool that helps the KE to check the conceptual model and to prepare the following acquisition meetings.

4.3 Experimentation Proposed Method

To prove the usefulness of GCM and its advantages versus the methods described, some experiments have been prepared, which are described in table 1. The GCM is compared with the KADS and IDEAL conceptualization stage. The experiments are carried out with two groups of people: without experience and with experience in conventional software. Within each group three models in three different domains are accomplished: take of decisions, diagnostic and interpretation. The objective is to prove the versatility of the method for several types of tasks and to see the influence of the KE experience in the use of any method. The experiment conditions are the following:

- Before the experiment begins, people who participate in the experiment attend a course where the method is explained.
- People have documentation where the method is described.
- A consultant is available for doubts.

Table 1. Experiments to study the method usefulness. Each row actually contains three experiments in three domains. Only the experiments labeled with an "x" have been completed.

KNOWLEDEDGE ENGINEER EXPERIENCE	METHOD	FINISHED
With experience in other software development fields	IDEAL	
	KADS	x
	GCM	x
Without experience	IDEAL	x
	KADS	
	GCM	x

Since two experiments have not been accomplished yet, definitive conclusions can not be drawn. However, the following facts have been observed:

- The detailed indication of the activities to accomplish reduces the initial "panic" among inexperienced people.
- That detailed list of activities focuses the attention in central points, so the time to elaborate the model is reduced.
- People who work with the proposed method in this work ask very concrete questions, about details. People who work with other methods ask more general questions. Sometimes they do not know what to do.
- Defining actions to check the model is considered very useful to detect mistakes

5 Conclusions

In this work a study of existing methods to model expertise has been presented. These methods are classified into two groups: works that propose specific resolution methods for each type of task and works that propose a general method for the elaboration of the conceptual model for any type of task. These methods have some deficiencies: there are not specific methods for all type of tasks and they are centered in what to do to describe a conceptual model, but not how to do it. That is, there is not a detailed guidance to build the model. They are not concerned with conceptual model evaluation aspects either.

The method proposed in this work presents following contributions:

- Detailed definition of activities to achieve a conceptual model.
- Evaluation aspects: detailed list of activities to check each knowledge level of the model and definition of two tools to validate the whole conceptual model integrating all knowledge levels: knowledge map and process model.

Besides, the method is being experimented rigorously. Until this moment the results are that the inexperienced people find less problems to build the model because of they have a guidance to work and to check the model. With this guide they feel more confident about what they are doing.

References

1. Ageenko I.I., SIGART Bulletin, Book Reviews. 9 (1), 1998, p. 345-47. Review of: Fensel D. The Knowledge Acquisition and Representation Language, KARL. Kluwer, 1995.
2. Angele, J., Fensel, D. and Studer, R., Domain and Task Modelling in MIKE. Proceedings of the IFIP WG 8.1/13.2 *Joint Working Conference, Domain Knowledge for Interactive System Design*, Geneva, Switzerland, May 8-10th, 1996.
3. Bylander T., Chandrasekaran B., Generic tasks for knowledge-based reasoning: the "right" level of abstraction for knowledge acquisition, *International Journal on Man-Machine Studies*, 26, 1987, p. 231-243.
4. Chandrasekaran B., Towards a Taxonomy of Problem Solving Types, *AI Magazine*, 4 (1), Winter/Spring 1983, p. 9-17.
5. Chandrasekaran B., Generic Task in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Desing, *IEEE Expert*, 1 (3), Fall 1986, p. 23-30.
6. Chandrasekaran B., Towards a Functional Architecture for Intelligence Based on Generic Information Processing Tasks, *Proceedings IJCAI*, Morgan Kaufmann, 1987, p. 1183-1192.
7. Chandrasekaran B., Design Problem Solving: A Task Analysis, *AI Magazine*, 11(4), Winter 1990, p. 59-71.
8. Chandrasekaran B., Johnson T.R. y Smith J.W., Task-Structure Analysis for Knowledge Modeling, *Communications of the ACM*, 35(9), September 1992, p. 124-137.
9. Clancey W. J., Heuristic Classification, *Artificial Intelligence*, 27, 1985, p. 289-350.
10. Gómez, A., Juristo, N., Montes, C., Pazos, J., *Ingeniería del conocimiento*, Centro de Estudios Ramón Areces S. A., Madrid, 1997.
11. Gómez, A., Moreno, A., Pazos, J. Sierra-Alonso, A. Knowledge maps: an essential technique for conceptualization, To appear in *Data & Knowledge Engineering*.
12. McDermott J., Preliminary Steps Towards a Taxonomy of Problem-Solving Methods, S. Marcus, *Automating Knowledge Acquisitiopn for Expert Systems*, Kluwer, 1988, p.225-266.
13. Molina, M., Gómez, A., Sierra, J.L. Reusable Components for Building Conventional and Knowledge-based Systems: The KSM Approach. *Proc. of the 9th International Conference on Software Engineering & Knowledge Engineering SEKE'97*, Madrid, 1997, p. 168-176.
14. Newell A., The Knowledge Level, *Artificial Intelligence*, 18(1), enero 82, p. 87-127.
15. Puerta A.R.; Tu S.W.; Musen M.A., Modeling tasks with mechanisms, *International Journal of Intelligent Systems*, 8(1), Jan. 1993, p. 129-52.
16. Rumbaugh J., Jacobson I., Booch G.: *The Unified Modelling Languaje Reference Manual*. Addison-Wesley, 1999.
17. Schreiber G., Wielinga B., Breuker, J. editores, *KADS: A Principled Approach to Knowledge-Based System Developement*, Academic Press, 1993.
18. Sierra-Alonso, A. *Lenguaje y proceso para construcción de modelos conceptuales expertos*. PhD Dissertation, Facultad de Informática, U. Politécnica de Madrid. To defend in 2000.
19. Steels L., Components of Expertise, *AI Magazine*, 4(1), summer 1990, p.28-49.
20. Studer R., Benjamins V.R., Fensel D., Knowledge Engineering: Principles and Methods, Data & Knowledge Engineering, 25(1-2), 1998, p. 161-197.

Segmenting the e-Commerce Market Using the Generative Topographic Mapping

A. Vellido[1]*, P.J.G. Lisboa[1], and K. Meehan[2]

[1]School of Computing and Mathematical Sciences, Liverpool John Moores University, Byrom St. Liverpool L3 3AF, UK
[2]Business School, Liverpool John Moores University, 98 Mount Pleasant, Liverpool L3 5UZ, UK

Abstract. The neural network-based Generative Topographic Mapping (GTM) (Bishop et al. 1998a, 1998b) is a statistically sound alternative to the well-known Self Organizing Map (Kohonen 1982, 1995). In this paper we propose the GTM as a principled model for cluster-based market segmentation and data visualization. It has the capability to define, using Bayes' theorem, a posterior probability of cluster/segment membership for each individual in the data sample. This, in turn, enables the GTM to be used to perform segmentation to different levels of detail or granularity, encompassing aggregate segmentation and one-to-one micro-segmentation. The definition of that posterior probability also makes the GTM a tool for fuzzy clustering/segmentation. The capabilities of the model are illustrated by a segmentation case study using real-world data of Internet users opinions on business-to-consumer electronic commerce.

1 Introduction

The process of mining a data set involves the discovery of those patterns of interest that might be present in it. One type of such patterns is the clusters in which the data points are grouped (Chen et al. 1996). In the context of Internet retailing, the identification of clusters of consumer types has been stated as "the most important use of data mining, as this type of information is useful in a myriad of other planning and development tasks" (Slater et al. 1999).

Cluster analysis is a central part of any market segmentation strategy and there have been recent calls for the design and deployment of market segmentation techniques grounded in a sound statistical framework (Wedel and Kamakura 1998). This paper proposes the Generative Topographic Mapping (GTM), a non-linear latent variable model devised and developed by Bishop et al. (1998a, 1998b), as a market segmentation tool that meets those needs with its data visualization and clustering capabilities. Presented as a principled alternative to the well-known Self-Organizing Map (Kohonen 1982, 1995), the GTM is shown in this study to accommodate segmentation strategies of different *granularity*, from micro-segmentation to the traditional aggregate segmentation. As a consequence, it gives the marketer the chance to reach a compromise between the heterogeneity of the data and the

* Corresponding author: Tel.: +44-0151-231-2188. Fax: +44-0151-207-4594. E-mail: A.Vellido@livjm.ac.uk

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 470—481, 2000.

managerial relevance of the segmentation solution. The GTM can also produce a probability of cluster/segment membership for each individual and, therefore, it can be used as a fuzzy clustering tool.

So far, only a few studies have addressed the problem of segmentation in the Internet context (McDonald 1996; Gordon and De Lima-Turner 1997; Vellido et al. 1999). Utilizing data from the *9th GVU's WWW Users Survey*, produced by the Graphics, Visualization & Usability Center (Kehoe et al. 1998), this study intends to address that gap whilst illustrating the aforementioned segmentation properties of the GTM. Part of them concerns itself with Internet users opinions of online shopping. The segmentation study will use, as a starting point, factors obtained from these data which have been shown in previous studies (Vellido et al. 2000) to be predictive of the propensity to buy on-line.

2 Description of the Data and Its Pre-processing

This study makes use of publicly available data from the web-based *9th GVU's WWW User Survey* (Kehoe et al. 1998). From the first two questions of its "*Internet Shopping (Part 1) Questionnaire"* (*"general opinion of using the WWW for shopping as compared to other means of shopping"* and *"opinion of Web-based vendors compared to vendors in other forms of shopping"*), 44 items were selected.

These items, in a previous study (Vellido et al. 2000), were subjected to factor analysis, for dimensionality reduction and latent variable exploration, and a nine-factor model was obtained following standard goodness-of-fit test procedures. The study intended to assess which factors were more predictive of the propensity to buy online. The data sample utilised contained records from 2180 Internet users and information (in the form of a binary dependent variable) of whether the respondent has or has not ever purchased online. Subsequently, a variable selection model, Automatic Relevance Determination (ARD) (MacKay 1995), associated to a Bayesian neural network, was applied to the resulting factor scores. The five most relevant factors were labelled according to previously published qualitative studies, and they are summarized in *table 1.*

FACTOR	DESCRIPTION	ATTRIBUTES
1	*Shopping experience: Compatibility*	*Control and convenience*
2	*Consumer risk perception / Environmental control*	*Trust and security*
3	*Affordability*	--
4	*Shopping experience: Effort*	*Ease of use*
5	*Shopping experience / Customer service*	*Effort / Responsiveness and empathy*

Table 1: Descriptive summary of the factors selected by the ARD model as predictive of online purchasing behaviour.

This previous study also showed that, from a hold-out sample, a supervised Bayesian neural network is able to correctly discriminate between the classes of purchasers and non-purchasers with a maximum of over 81% accuracy.

The practice of factor analysing the observable data, followed by the clustering of the obtained factor scores is known, in the marketing literature, as the *tandem approach* towards segmentation. Factor analysis helps to overcome the limitations associated with survey-based segmentation studies, i.e. noisy data, poorly measured variables based on subjective ratings, and unbalanced item selection across the domain of the surveyed constructs (Green and Krieger 1995) It also makes the interpretation (labelling) of the factor structure possible. Some authors have argued against the tandem approach on the basis that it might discard relevant information and distort the true cluster structure of the data (Arabie and Hubert 1994). Green and Krieger (1995) and Schaffer and Green (1998) acknowledge that a good validation of the segmentation results might stem from the relation of alternative clusterings to some profit-based measure. This provides a justification for our use of the *tandem approach*: in this study, the segmentation is carried out according to factors which have been shown to discriminate, quite accurately, the classes of *purchasers* and *non-purchasers,* each of which offers a profit/non-profit opportunity to the marketer. The GTM is expected to generate clusters that discriminate between consumers that belong to each of the classes defined by the dependent variable. This property has been observed in the SOM model (Vellido et al 1999; Serrano-Cinca 1996). The resulting cluster solution has an explicitly profit-based interpretation, as each cluster can be described and targeted in a marketing campaign with the combined knowledge of the latent factors that shape it and the propensity to buy which they predict.

3 The GTM as a Principled Model for Data Visualization and Market Segmentation

The Generative Topographic Mapping (GTM) (Bishop et al. 1998a, 1998b) is a non-linear latent variable model that generates a probability density in the multi-dimensional data space, using a set of latent variables of smaller dimension. This non-linear mapping is described by the generalized linear regression model

$$\mathbf{y} = \mathbf{W}\phi(\mathbf{u}), \tag{1}$$

where $\mathbf{u}$ is an L-dimensional vector of latent variables, $\mathbf{W}$ is the matrix that generates the explicit mapping from latent space to an L-dimensional manifold embedded in data space, and ϕ is a set of R basis functions which, in this study, are chosen to be Gaussians. For the non-linear mapping to remain analytically and computationally tractable, and also to elaborate a principled alternative to the Self-Organizing Map (SOM) (Kohonen 1982, 1995), the prior distribution of $\mathbf{u}$ in latent space is defined as a discrete grid, similar in spirit to the grid of the SOM

$$p(\mathbf{u}) = \frac{1}{M}\sum_{i=1}^{M}\delta(\mathbf{u} - \mathbf{u}_i) \quad , \tag{2}$$

where M is the number of its nodes. Since the data do not necessarily lie in an L-dimensional space, it is necessary to make use of a noise model for the distribution of data vectors. The integration of this data distribution over the latent space distribution, gives

$$p(\mathbf{x}|\mathbf{W},\beta) = \int p(\mathbf{x}|\mathbf{u},\mathbf{W},\beta)p(\mathbf{u})d\mathbf{u} = \frac{1}{M}\sum_{i=1}^{M}\left(\frac{\beta}{2\pi}\right)^{\frac{D}{2}} \exp\left\{-\frac{\beta}{2}\left\|\mathbf{m}_i - \mathbf{x}\right\|^2\right\}, \quad (3)$$

where D is the dimensionality of the input space, and $\mathbf{m}_i = \mathbf{W}\phi(\mathbf{u}_i)$ for the discrete node representation, according to (1). Using the SOM terminology, $\mathbf{m}_i$ can be considered as *reference vectors*, each of them the centre of an isotropic Gaussian distribution in data space (Bishop et al. 1998b). The log-likelihood can now be defined as

$$L(\mathbf{W},\beta) = \sum_{n=1}^{N} \ln\left[p\left(\mathbf{x}^n|\mathbf{W},\beta\right)\right] \quad (4)$$

for the whole input data set $\{\mathbf{x}^n\}$.

The distribution (3) corresponds to a constrained Gaussian mixture model (Hinton et al. 1992), hence its parameters, $\mathbf{W}$ and β, can be determined using the Expectation-Maximization (EM) algorithm (Dempster et al. 1977), details of which can be found in Bishop et al.(1998a). As part of the Expectation step, the mapping from latent space to data space, defined by (1), can be inverted using Bayes' theorem so that the posterior probability of a GTM node *i,* given a data-space point is defined as

$$R_i^n \equiv p\left(\mathbf{u}_i\left|\mathbf{x}^n\right.\right) = \frac{\exp\left[-\frac{\beta}{2}\left\|\mathbf{m}_i - \mathbf{x}^n\right\|^2\right]}{\sum_{i'}^{M}\exp\left[-\frac{\beta}{2}\left\|\mathbf{m}_{i'} - \mathbf{x}^n\right\|^2\right]} \quad (5)$$

This is known as the *responsibility* taken by each node *i* for each point *n* in the data space. Each of the nodes in latent space can be interpreted, as it is for the SOM model (Ripley 1996), as a cluster or, in the context of market segmentation, as a segment. Therefore, equation (5) defines a posterior probability of cluster membership for each cluster and each data point. Assuming that each data point / individual belongs only to one cluster (non-overlapping clustering) but the information it conveys does not suffice to uniquely assess its cluster membership (Wedel and Kamakura 1998; McLachlan and Basford 1988), the GTM can then be used as a *fuzzy-clustering* tool. This posterior probability also provides the model with the data visualization capabilities that other models that project high-dimensional data into a visualization space possess. The mapping defined by (1) ensures the insightful property of *topographic ordering*: any two points that are close in latent space will be mapped to points that are necessarily close in data space. Consequently, neighbouring clusters in latent space correspond to groups of data that are also close in data space. This preservation of the topographic order, naturally brings about another feature of the GTM which is most relevant for its implementation as a market segmentation tool: besides the definition of each node in latent space as a cluster / segment, these nodes can be aggregated in a principled way to form macro-clusters. This aggregation methodology is described in section 4.2.

Therefore, the GTM provides a way to address the always contentious issue of the level of detail or *granularity* with which the segmentation should be carried out. This is the core of the problem of "continuous distribution of heterogeneity versus market segments" (Wedel and Kamakura 1998) that confronts two different views. The first being where the heterogeneity amongst consumers is not so pronounced that the

possibility of grouping them into segments is rejected. The second would consider the partition of the consumer representation continuum into segments as an artefact (Allenby and Lenk 1994), thus assuming that markets are perfectly heterogeneous. The latter is the basis for any attempt to define models of *one-to-one* marketing or *finer* segmentation (Kara and Kaynak 1997; Firat and Shultz 1997). This capability of the GTM is of paramount importance in e-commerce, because the on-line marketer can "see the opportunity to increase the consumer's satisfaction and the effectiveness of marketing efforts by learning their behaviour and being able to pinpoint each individual's preferences and adapt their offerings accordingly" (Wallin 1999).

4 Experiments

This section gives a brief account of the implementation of the GTM, focusing on model selection issues, followed by the presentation and discussion of the segmentation results.

4.1 Implementation of the GTM model

The complexity of the mapping generated by the GTM model is mainly controlled by the number and form of the basis functions. Further control of this effective complexity can be achieved with the addition of a regularization term to the error function in such a way that the training of the GTM would consist of the maximization of a *penalized* log-likelihood

$$L_{PEN}(\mathbf{W},\beta) = \sum_{n=1}^{N} \ln p(\mathbf{x}_n|\mathbf{W},\beta) + \frac{1}{2}\alpha\|\mathbf{w}\|^2, \quad (6)$$

where $\mathbf{w}$ is a vector shaped by concatenation of the different column vectors of the weight matrix $\mathbf{W}$ and α is a real-valued regularization coeficient. This regularization term is effectively preventing the GTM to fit the noise in the data and is used under the assumption that there exist an underlying data generator which is a combination of the density functions for each of the segments.

The optimum values for all these parameters should ideally be evaluated in a continuous space of solutions. Given that the GTM is formulated within a probabilistic framework, this can be accomplished using the Bayesian formalism and, more specifically, the evidence approximation (MacKay 1992). The application of this methodology (Bishop et al. 1998b) produces update formulae for the regularization coefficient α and for the inverse variance of the noise model β.

The model by which we illustrate the use of the GTM consists of a grid of 5x5 basis functions with a common width $\sigma = 1$. The GTM was trained with a class-balanced data set of 779 individuals, and the values for the regularization coefficient and the inverse variance of the noise model after training were, in turn, $\alpha = 1.12$ and $\beta = 1.57$. A fixed grid in latent space of 15x15 nodes was selected as a compromise on the level of detail or *granularity* of the cluster / segment solution.

4.2 Segmentation results

The equation (5) provides a posterior probability of cluster membership for each individual. In order to visualize that probability for a complete set of data, the information has to be somehow summarized. This can be done (Bishop et al. 1998a) by calculating the *mode* of the distribution,

$$i^{\max} = \arg\max_{\{i\}} R_{in} \quad , \tag{7}$$

The information is now visualized in the way that is usual with the SOM model (Kohonen 1995).

Figure 1 represents the whole data set by the *mode* (7), as mapped into the trained GTM. Each individual has been assigned a colour, black or white, depending on its *true* class membership as described in section 2. Grey nodes indicate that individuals from different classes have been mapped into them.

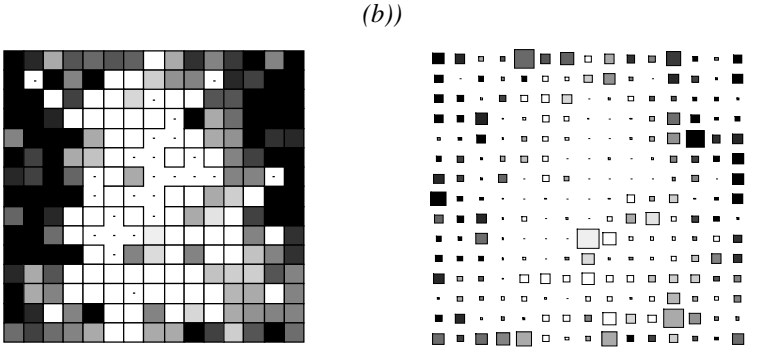


Fig. 1: GTM class-membership maps. All the individuals in the training sample are mapped onto the nodes in the latent visualization space that correspond to their *modes* as defined by (7): a) Each of them has been shaded according to its class-membership: white for *purchasers*, black for *non-purchasers*. Those GTM nodes with more than one pattern mapped onto them are depicted in shades of grey, corresponding to the class-membership proportion. b) The same pseudo-colour representation has been used, but now the relative size of the squares/nodes corresponds to the number of patterns mapped onto them. Regions of high pattern occurrence in latent space can thus be visualized.

Figure 1(a) shows that the GTM, *without any prior information on class membership* has managed to separate, quite clearly, the classes of *purchasers* and *non-purchasers*. Such result shows that our semi-predictive approach, using factors shown to predict the propensity to buy on-line as the basis for segmentation, is an appropriate methodology for market profit optimization.

Figure 1(b) conveys information about the size of each of the nodes / micro-clusters (number of individuals whose *mode* corresponds to each of them) showing agglomerations of data in specific areas. This provides a starting point for the definition of the macro-clusters of an aggregate segmentation strategy.

Lewis *et alia* (1997) propose, for the SOM model, to designate as macro-cluster centroids those nodes with the largest number of patterns mapped onto it. Each of the remaining nodes will be added to the macro-cluster with the closest centroid.

An example of the combination of these methods is the 9-segment solution in figure 2. Those segments have to be interpreted in terms of the segmentation bases. For that purpose and, as defined in section 3, the *reference vectors* $\mathbf{m}_i = \mathbf{W}\phi(\mathbf{u}_i)$ in data/bases space, for each node i in latent space, will be used. Each of the variables shaping these vectors can be visualized using a *reference map*, which is a pseudo-colour representation of its numerical values. The reference maps for our trained GTM are shown in figure 3.

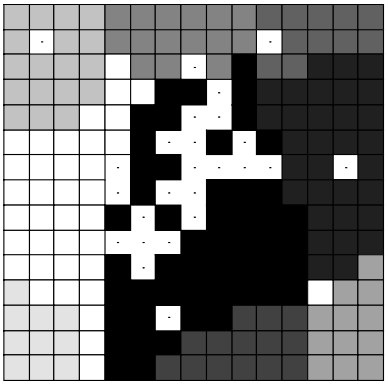


Fig. 2*:* A 9-segment solution obtained with the clustering procedure described in the text. It is interpreted in *table 3* with the help of the reference maps in *figure 3.*

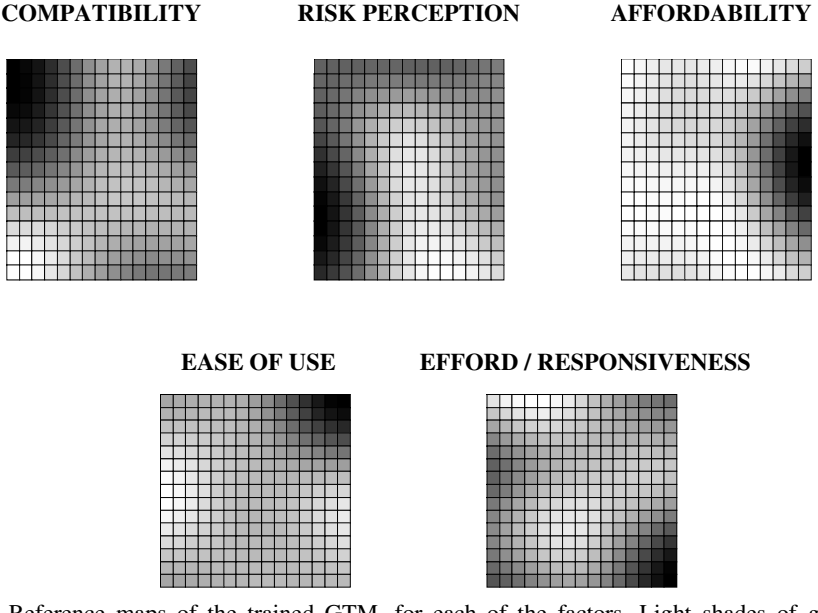


Fig. 3: Reference maps of the trained GTM, for each of the factors. Light shades of grey correspond to high values of the elements of the reference vectors, whereas dark colours correspond to low values. The interpretation of those high and low values is reported in *table 2.*

The meaning of the numerical values (white for high, black for low in a grey-shaded palette) is described in table 2 and the interpretation of the 9 segments according to the reference maps, its labelling and its size, are summarized in table 3.

Table 2: Interpretation of the low and high (negative / positive) values of the factor scores (inputs to the model) and reference vectors of the trained GTM.

	POSITIVE VALUES	**NEGATIVE VALUES**
Factor 1 *Compatibility: Control / Convenience*	People who perceive shopping on the WWW as compatible and convenient, feeling they are in control of the shopping process.	People who do not perceive shopping on the WWW as either compatible or convenient, or do not feel in control of the shopping process.
Factor 2 *Risk perception: Trust / Security*	People who do not perceive that shopping online entails a differential risk. They find online vendors trustworthy.	People who perceive security and economic risks as major deterrents to shop online. They find online vendors untrustworthy.
Factor 3 *Affordability*	People who find the whole prerequisites for online shopping as affordable.	People who find the whole prerequisites for online shopping as unaffordable.
Factor 4 *Shopping Experience: Effort / Ease of use*	People who find shopping online as a complicated undertaking, difficult to learn, that requires a lot of mental effort.	People who find shopping online easy and unproblematic.
Factor 5 *Shop.Exp.;Customer Service Effort / Responsiveness*	People who consider that online vendors provide a responsive customer service, reducing the effort involved in the online shopping process.	People who do not consider that online vendors provide a responsive customer service.

Table 3: Description of the 9-segment solution proposed in the text. Segments are numbered, according to *figure 2,* in the first column of the table and described in the second column according to *table 2.* The relative size and proposed label of each segment are included in the third column.

	SEGMENT DESCRIPTION	**LABEL and SIZE (%)**
1	All factors present medium-to-high values. Values of *Perception of risk, Affordability* and *Effort/Responsiveness* are specially high. They are mainly on-line *purchasers.*	*Convinced* (22.7%)
2	Most factors present medium values except *Affordability*, which is very low. They are mainly *non-purchasers*.	*Cost conscious* (16.7%)
3	High values of *Affordability* and *Perception of risk*, but rather low values of *Effort/responsiveness*. Mainly *purchasers.*	*Security and cost confident* (9.4%)
4	Low *Compatibility* and very low *Ease of use.* Medium to high *Affordability*. Mainly *non-purchasers*.	*Complexity avoiders* (5.9%)
5	Similar to the segment 7 (*Unconvinced*) but scoring higher in the factor of *Compatibility*. This brings about a mixture of *purchasers* and *non-purchasers.*	*Undecided* (9.6%)
6	This small group scores very low in *Effort/responsiveness*, but medium-to-high in the rest of factors. Mixed *purchasers* and *non-purchasers*.	*Customer service wary* (5.6%)
7	Very low on *Compatibility* and rather low on *Perception of risk,* although rather high on *Affordability*. Mainly *non-purchasers.*	*Unconvinced* (8.9%)
8	High *Compatibility* compounded with low values of *Perception of risk*. Rather *non-purchasers*.	*Security conscious* (5.3%)
9	High on *Affordability* and *Ease of Use*, but low in all the risk related factors. Mainly *non-purchasers.*	*Risk avoiders* (15.9%)

The GTM can also be used to discriminate *segments of one*, otherwise referred to as micro-segmentation, one-to-one-segmentation or finer-segmentation (Kara and Kaynak 1997), by augmenting the latent-space grid to a sufficient resolution. The resulting segments can be characterized using the *reference maps* as described above. Nevertheless, it has to be borne in mind that segments do not naturally arise from the data description of the market, and there is always a degree of subjectivity regarding its definition, motivated by the need to obtain a market description of managerial relevance. The characteristics of the segments have to be adapted to accommodate a trade off between the costs of the segmentation and the efficiency and likelihood of consumer response (Wind 1978; Wedel and Kamakura 1998).

4.3 Segment profiling

There are several criteria that define the feasibility of a market segmentation strategy. Amongst others, *substantiality* and *actionability*, which are met by the segment solution provided in this study. Another two criteria, *identifiability* and *accesibility* might sometimes depend on the availability of secondary information, such as demographic and socio-economic data. This type of information has been shown to be neither the most effective in developing segments (Wedel and Kamakura 1998) nor a good predictor of the propensity to buy on-line (Vellido et al. 2000). Nevertheless, they may sometimes be the only kind of data available to the marketer.

We profile now the 9-segment solution provided in the previous section, making use of the following bases: *age, household income, years of Internet experience, average of hours a week of Internet usage* and *gender*. This selection of bases is by no means exhaustive. Bearing in mind that this is just an illustration of a procedure that ultimately depends on the availability of secondary information and for the sake of brevity, the profiling, summarized in table 4, will be limited to the aforementioned variables.

Many conclusions can be drawn from these results. Table 4, though, is quite self-explanatory, so that only some general ideas will be sketched here. Segments 1 and 3 (*Convinced* and *Security and Cost Conscious*), mainly composed of *purchasers*, characterised as mostly male in their late twenties to early forties, with high-band income and Internet-savvy. Most of the segments with majority of *non-purchasers* tend to be dominated by females. An exception to this is segment 4 (*Complexity Avoiders*) who are largely people with very low Internet usage and experience. Segment 2 (*Cost Conscious*) add rather lower incomes to that lack of experience. Segments 5 and 8 (*Undecided* and *Security Conscious*) are also rather similar, with a young profile and average-to-low Internet usage and experience. Their main difference lies in the income distribution, which loads higher in the extreme sub-segments (low and high) in the case of segment 5. Segment 6 (*Customer Service Wary*) is again rather young and combines low Internet usage and experience with low income. We might interpret segment 7 (*Unconvinced*) as a variant of the previous that is strongly female and also including a very significant mature sub-segment. Finally, segment 9 (*Risk avoiders*) seems to agree notably, aside of a higher female proportion, with the average characteristics of the sample as a whole.

Table 4*:* The bases *age, average of hours a week of Internet usage, years of Internet experience, household income* (in US $) and *gender* are represented, in the left-side of the table, as (A), (H), (Y), (I) and (G). All the figures in the table, corresponding to those 5 bases, are percentages of the segment size (or the sample size in the case of the first column). All the segment sizes and the percentage of the sample size they represent are presented in the last row of the table.

		All	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**
1(A)	**0A ≤ 25**	26.1	19.8	28.5	17.8	32.6	34.7	31.8	29.0	31.7	24.2
	25 <A ≤35	25.2	30.5	20.7	30.1	8.7	30.6	25.0	17.4	24.4	27.4
	35 <A ≤45	23.4	23.7	20.0	28.8	39.1	17.4	25.0	14.5	19.5	26.6
	A > 45	25.3	26.0	30.8	23.3	19.6	17.3	18.2	39.1	24.4	21.8
0(H)	H ≤ 10	34.1	22.0	43.8	21.9	50.0	36.0	38.6	36.2	46.3	34.7
	10 <H ≤20	31.9	29.4	32.4	30.2	32.6	34.7	38.7	37.7	31.7	28.2
	H > 20	34.0	48.6	23.8	47.9	17.4	29.3	22.7	26.1	22.0	37.1
1(Y)	Y ≤ 1	18.2	6.2	28.5	4.1	30.4	14.7	31.8	24.6	24.4	20.2
	1 <Y ≤3	38.2	39.0	44.6	31.5	34.8	42.6	36.4	36.3	36.6	34.6
	3 <Y ≤5	22.4	22.0	18.4	34.3	17.4	22.7	15.9	17.4	24.4	26.7
	Y > 5	21.2	32.8	9.5	30.1	17.4	20.0	15.9	21.7	14.6	19.5
(I)	I ≤ 30K	26.1	17.5	33.1	20.5	26.1	29.3	34.1	36.2	19.5	25.8
	30K<I ≤50K	28.1	25.4	31.5	24.7	26.1	24.0	31.8	21.8	36.6	33.1
	50K<I ≤70K	18.8	25.5	13.1	16.4	19.5	17.4	13.6	20.3	26.8	16.1
	I > 70K	27.0	31.6	22.3	38.4	28.3	29.3	20.5	21.7	17.1	25.0
2(G)	Male	51.6	65.0	44.6	65.8	54.3	48.0	40.9	36.2	46.3	46.8
	Female	48.4	35.0	55.4	34.2	45.7	52.0	59.1	63.8	53.7	53.2
	0Segment **size** *%*	779 *100*	177 *22.7*	130 *16.7*	73 *9.4*	46 *5.9*	75 *9.6*	44 *5.6*	69 *8.9*	41 *5.3*	124 *15.9*

5 Conclusion

For the on-line marketers, data mining is "an integral part of the ongoing business process whereby they ensure that their site remains successful and responsive to user and customer needs" (Slater et al. 1999)

The neural network-inspired GTM, as a non-linear latent variable model based on a constrained mixture of Gaussians, whose parameters are estimated with the EM algorithm, is defined within a sound statistical framework. Referring generally to mixture of distribution models, Wedel and Kamakura (1998, p.33) argue that "the statistical approach clearly is a major step forward in segmentation research." That is in contrast with traditional non-overlapping clustering algorithms used for segmentation, grouped in two main types: *hierarchical* (e.g. Ward's method) and *nonhierarchical* (e.g. K-means). These algorithms, as well as other neural network-

based model such as the SOM, are limited by its heuristic nature and their results can not be justified by standard statistical theory.

Neural networks have been quoted to be promising models for personalization and mass-customization of the e-commerce market (Scharl and Brandtweiner 1998; Wallin 1999). The GTM, as a neural network-inspired model, has been shown to provide a principled framework for one-to-one segmentation, but also for the traditional aggregate segmentation. It has also been shown that this model can be used as a fuzzy clustering tool.

Its utilization for the analysis of data concerning consumers' opinions on online shopping has revealed several interesting features in these data. Firstly, it has been shown that the GTM has discriminated, to a high degree, the classes of on-line purchasers and non-purchasers *without any a priori information* of the class membership of the individuals in the sample. This reinforces similar results obtained with the SOM model (Vellido et al. 1999). Secondly, we have provided an example of the use of the GTM for aggregate segmentation, including the interpretation and profiling of the segments obtained.

References

Allenby, G.M. and Lenk, P.J.: Modelling household purchase behaviour with logistic normal regression. *Journal of the American Statistical Association,* 89(December), (1994) 1218-1231.

Arabie, P., and Hubert, L.: Cluster analysis in market research. In Bagozzi, R.P. (Ed.), *Advanced methods in marketing research.* Oxford: Blackwell & Company, (1994) 160-189.

Bishop, C.M., Svensén, M., and Williams, C.K.I.: GTM: the Generative Topographic Mapping. *Neural Computation,* 10(1), (1998) 215-234.

Bishop, C.M., Svensén, M. and Williams, C.K.I.: Developments of the Generative Topographic Mapping. *Neurocomputing* 21(1-3), (1998) 203-224

Chen, M.S., Han, J. and Yu, P.S.: Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering,* 8(6) (1996) 866-884.

Dempster, A.P., Laird, N.M., and Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B,* 39(1), (1977) 1-38.

Firat, A.F., and Shultz II, C.J.: From Segmentation to fragmentation: Markets and marketing strategy in the postmodern era. *European Journal of Marketing,* 31(3-4), (1997) 183-207.

Gordon, M.E., and De Lima-Turner, K.: Consumer attitudes towards Internet advertising: A social contract perspective. *International Marketing Review,* 14(5), (1997) 362-375.

Green, P.E., and Krieger, A.M.: Alternative approaches to cluster-based market segmentation. *Journal of the Market Reseach Society,* 37(3), (1995) 221-239.

Hinton, G.E., Williams, C.K.I., and Revow, M.D.: Adaptive elastic models for hand-printed character recognition. In Moody, J.E., Hanson, S.J., and Lippmann, R.P. (Eds.) *Advances in Neural Information Processing Systems,* Vol.4, pp.512-519. Morgan Kauffmann (1992).

Kara, A., and Kaynak, E.: Markets of a single customer: exploiting conceptual developments in market segmentation. *European Journal of Marketing,* 31(11-12), (1997) 873-895.

Kehoe, C., Pitkow, J., and Rogers, J.D.: *9th GVU's WWW user survey* (1998).
URL: http://www.gvu.gatech.edu/user_surveys/survey-1998-04/

Kohonen, T.: Self-organized formation of topologically correct feature maps. *Biological Cybernetics,* 43(1), (1982) 59-69.

Kohonen, T.: *Self-Organizing Maps.* Berlin: Springer-Verlag (1995).

Lewis, O.M., Ware, J.A. and Jenkins, D.: A novel neural network technique for the valuation of residential property. *Neural Computing & Applications,* 5(4) (1997) 224-229.

MacKay, D.J.C.: A practical Bayesian framework for back-propagation networks. *Neural Computation,* 4(3) (1992) 448-472.

MacKay, D.J.C.: Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks network-computation in neural systems. *Network: Computation in Neural Systems,* 6, (1995) 469-505.

McDonald, W.J.: Internet customer segments: an international perspective. In Droge, C., & Calantone, R. (Eds.), *Enhancing knowledge development in marketing.* Chicago, IL: American Marketing Association, (1996) 338-344.

McLachlan, G.J. and Basford, K.E.: *Mixture Models: Inference and Applications to Clustering.* New York: Marcel Dekker (1988).

Ripley, B.: *Pattern recognition and neural networks.* Cambridge University Press (1996).

Schaffer, C.M., and Green, P.E.: Cluster-based market segmentation: some further comparisons of alternative approaches. *Journal of the Market Research Society,* 40(2) (1998) 155-163.

Scharl, A. and Brandtweiner, R.: A conceptual research framework for analyzing the evolution of electronic markets. *Electronic Markets Newsletter,* 8(2) (1998) 1-6.

Serrano-Cinca, C.: Self-organizing neural networks for financial diagnosis. *Decision Support Systems.* 17, (1996) 227-238.

Slater, D., Mulvenna, M., Büchner, A. and Moussy, L.: Mining marketing intelligence from Internet retailing data: user requirements & business process description. In *Proceedings of the European Multimedia, Microprocessor Systems and Electronic Commerce (EMMSEC'99) Annual Conference.* Stockholm, Sweden (1999)

Vellido, A., Lisboa, P.J.G. & Meehan, K.: Segmentation of the on-line shopping market using neural networks. *Expert Systems with Applications,* 17(4) (1999)

Vellido, A., Lisboa, P.J.G. & Meehan, K.: Characterizing and segmenting the business-to-consumer e-commerce market using neural networks. To appear in Lisboa, P.J.G., Vellido, A. and Edisbury, B. (Eds.): *Neural Networks Mean Business.* Singapore: World Scientific (2000)

Wallin, E.O.: Consumer personalization technologies for e-commerce on the Internet: a taxonomy. In *Proceedings of the European Multimedia, Microprocessor Systems and Electronic Commerce (EMMSEC'99) Annual Conference*. Stockholm, Sweden (1999)

Wedel, M. and Kamakura, W.A.: *Market Segmentation. Conceptual and Methodological Foundations.* International Series in Quantitative Marketing. Massachusetts: Kluwer Academic Publishers (1998).

Wind, Y.: Issues and advances in segmentation research. *Journal of Marketing Research,* 15(August), (1978) 317-337.

Designing a Fast Neuro-fuzzy System for Speech Noise Cancellation

Anna Esposito[1], Eugène C. Ezin[1,2], and Carlos A. Reyes-Garcia[1,3]

[1] International Institute for Advanced Scientific Studies (IIASS)
Via G. Pellegrino 19, 84019, Vietri sul Mare, SA, Italy, iiassae@tin.it
[2] Institut de Mathématiques et de Sciences Physiques (IMSP)
B.P. 613 Porto-Novo, Benin, e_ezin@hotmail.com
[3] Instituto Tecnologico de Apizaco-COSNET
Apizaco, Tlaxcala, Mexico 90300, kargaxxi@yahoo.com

Abstract. Noise canceling is an adaptive interference filtering technique that has shown to be highly adventageous in many applications where fixed filters are not efficient. We present an experimental neuro-fuzzy inference system, based on the ANFIS architecture, which has been implemented with the objective to perform nonlinear adaptive noise cancellation from speech. The novelty of the system described in the present paper, with respect to our previus work, consists in a different set up, which requires two inputs with seven membership functions each, and uses a second order *sinc* function to generate the nonlinear distortion of the noise. This set up allows a better generalization to the system for learning the noise features. Indeed, the system was trained only once during few epochs, with a sample of babble noise, but it was able to clean speech sentences corrupted not only with the same noise, but also with car, traffic, and white noise. The average improvement, in terms of SNR, was 37 dB without further training, resulting in a great reduction of the computational time.

Keywords: Neuro-Fuzzy System, ANFIS, Adaptive Noise Cancellation, Signal to Noise Ratio, Passage Dynamics.

1 Introduction

One of the main requirements to achieve efficient Automatic Speech Recognition (ASR) is to have the speech signal to be processed, as clean as possible. Many efforts have been done in order to eliminate the noise from the speech wave, or to reduce its negative influence during the processing and recognition stages. The general approach is to pass the noisy signal through a sort of filter, which will try to get rid of the noise while leaving unchanged the information signal. The filters might be fixed or adaptive. The design of fixed filters requires previous knowledge of both signal and noise, whereas adaptive filters require little or no knowledge in advance, since they have the ability to dynamically adjust their own parameters according to the characteristics of the noise being processed.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 482–492, 2000.

Noise Cancellation is an adaptive filtering technique, which has shown to be highly advantageous, in many applications. To avoid the need of prior knowledge, the method requires a reference input which is supplied by one or more sensors located in the noise field where the information signal is weak or undetectable. Linear adaptive noise cancellation has been successfully applied to many real world applications [1,2,3]. Further research has expanded the concept to the realm of nonlinear adaptive systems [4,5,6,7,8,9].

In Nonlinear Adaptive Noise Cancellation Systems, the goal is to identify a model of the unknown nonlinear dynamics obtained by the passage of noise through different environments. Once this model is identified, it is possible to filter out the interference from a detected signal. One system with the capability to adjust itself to changing patterns is ANFIS [9,10,11,12], which also is able to implement a nonlinear mapping from the input to the output space. ANFIS is a neuro-fuzzy system that combines the learning capabilities of neural networks, with the functionality of fuzzy inference systems [9,13].

For simulating the characteristics of the passage dynamics the noise goes through, we applied a nonlinear second order *Sinc* function to generate the intercepting noise wave. For training and testing we build up a database of 22 sentences of real speech, spoken in Italian and American English, which during the experiments were corrupted with four different types of recorded noise (traffic, babble, car, and white noise). The resulting noisy sentences were used as input to the ANFIS system. In this paper we describe the set of initial parameters which allowed the system to attain very good performance when cleaning the noisy sentences, giving an average SNR improvement of 37.14 dB.

2 Nonlinear Adaptive Noise Cancellation

The objective of nonlinear adaptive noise cancellation (NANC) is to filter out an interference component by identifying a nonlinear model between a measurable noise source and the corresponding not measurable interference.

In general, there is a known noise source $n(t)$ which goes through an unknown nonlinear passage, which transforms it into a distorted and delayed noise $d(t)$. The distorted noise comes added to the speech signal $x(t)$, forming the detected signal $y(t)$:

$$y(t) = x(t) + d(t) = x(t) + f(n(t), n(t-1)) \tag{1}$$

where

$$f(n(t), n(t-1)) \tag{2}$$

describes the dynamic passage through which the noise wave goes to corrupt the speech signal. The main problem is to cancel the distorted noise $d(t)$ from the received signal $y(t)$ for getting the cleaned spoken signal $x(t)$. Since the noise signal $n(t)$ goes through passage dynamics, which are not measurable beforehand and vary with time by the changing environment, it is necessary to estimate $d(t)$ before being able to cancel it from the received signal $y(t)$.

To estimate $d(t)$, and to get the clean version of the speech signal we used the ANFIS architecture described below. During the training, the ANFIS system

outputs an estimation $\widehat{d}(t)$ of the distorted noise, which will be tuned up by comparing it to $y(t)$, which is taken as the desired output, since it contains the real distorted noise, together with the real speech signal [9]. To update the system parameters the learning procedure goes through two phases: a forward and a backward phase. In the forward phase the data flow until layer 4 where the **consequent parameters** are identified applying the least-square estimator. Then the outputs of layer 4 are combined in layer 5 and the squared error is calculated as the squared difference between the target $y(t)$ and the net output $\widehat{d}(t)$. In the backward phase, the error rate is propagated from the net output to the input end and the **premise parameters** in layer 1 are updated using the backpropagation algorithm [9,10,11,12].

2.1 Passage Dynamics Function

The passage dynamics can be defined as the delay and the distortion introduced in the noise wave by the transmission channels it goes through before corrupting the speech signal. The passage dynamics function takes as input the noise signal and outputs the distorted and delayed interference signal. Each type of function is supposed to simulate a particular kind of physical environment where the noise passes through. In the present report we selected the following nonlinear *sinc* function of order 2 represented below

$$d(t) = f(n(t), n(t-1)) = 4\frac{\text{sinc}\,(n(t)) \cdot n(t-1)}{1 + [n(t-1)]^2} \tag{3}$$

This function was chosen after comparing the system's performance against those obtained using the second and third order *sin* functions, and the third order *sinc* function.

3 ANFIS Configuration

ANFIS is a hybrid learning architecture, proposed by Jang [9,10,11,12]. As an adaptive network, the parameters in some ANFIS nodes are adapted during training (in such case the node is called **adaptive** and it is graphically represented by a square). There are also nodes whose parameters remain unchanged during training (they are called **fixed** and are represented by a circle). The network architecture we implemented is composed of five layers.

– The first layer of our configuration takes as inputs the noise wave $n(t)$ and its delayed version $n(t-1)$. For this implementation we used **seven** trapezoidal membership functions for each input, whereas in previous implementations [15,16] we were using only two. The reason for using such a number of membership functions comes from the hypothesis that, a greater number of linguistic properties will give a more detailed description of the noise wave. The linguistic properties assigned to each input variable, for our purposes, were respectively *VeryLow (VL), Low(L), More or Less Low(MLL), Medium*

(M), More or Less High (MLH), High (H) and *VeryHigh (VH).* Therefore the first layer has 14 adaptive nodes whose node function is represented by

$$\begin{array}{rc} O_{1,1} = & \mu_{VL_1}(n(t)) \\ O_{1,2} = & \mu_{L_1}(n(t)) \\ \vdots & \vdots \\ O_{1,14} = & \mu_{VH_2}(n(t-1)) \end{array} \tag{4}$$

– In the second layer there are 49 fixed nodes whose output is the product (*prod*) of all the incoming signals:

$$\begin{array}{rc} O_{2,1} = w_1 = & prod[\mu_{VL_1}(n(t)), \mu_{VL_2}(n(t-1))] \\ O_{2,2} = w_2 = & prod[\mu_{VL_1}(n(t)), \mu_{L_2}(n(t-1))] \\ \vdots & \vdots \\ O_{2,49} = w_{49} = & prod[\mu_{VH_1}(n(t)), \mu_{VH_2}(n(t-1))]\,. \end{array} \tag{5}$$

– The third layer also has 49 fixed nodes, whose output is computed as the ratio of the respective node output in the previous layer and the sum over all the outputs in the previous layer:

$$\begin{array}{rc} O_{3,1} = \overline{w}_1 = & \frac{w_1}{w_1+w_2+\cdots+w_{49}} \\ O_{3,2} = \overline{w}_2 = & \frac{w_2}{w_1+w_2+\cdots+w_{49}} \\ \vdots & \vdots \\ O_{3,49} = \overline{w}_{49} = & \frac{w_{49}}{w_1+w_2+\cdots+w_{49}} \end{array} \tag{6}$$

Each node in this layer is directly connected with only one node in the next layer.

– In the fourth layer there are 49 adaptive nodes whose node function is

$$\begin{array}{rcl} O_{4,1} = \overline{w}_1 f_1 = & \overline{w}_1(p_1 n(t), q_1 n(t-1), r_1) \\ O_{4,2} = \overline{w}_2 f_2 = & \overline{w}_2(p_2 n(t), q_2 n(t-1), r_2) \\ \vdots & \vdots \\ O_{4,49} = \overline{w}_{49} f_{49} = & \overline{w}_{49}(p_{49} n(t), q_{49} n(t-1), r_{49}) \end{array} \tag{7}$$

– The fifth layer has a single fixed node which computes the overall output as the sum of all the incoming signals:

$$O_{5,1} = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{8}$$

This network is functionally equivalent to a 49 rules Sugeno type Fuzzy Inference System [9,13] .

3.1 Network Configuration

During the experiments, we tested several parameter settings, as well as several kinds and several number of fuzzy membership functions. Through a heuristic

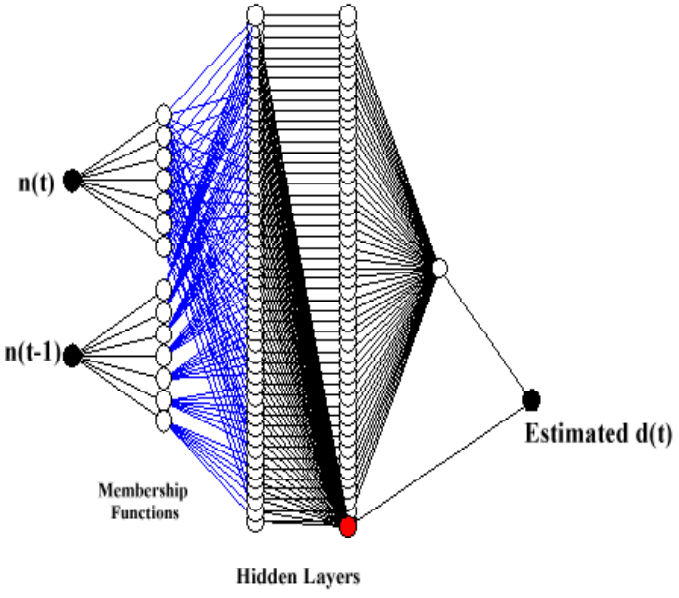


Fig. 1. Network Configuration.

selective process, we ended up with seven *trapezoidal* membership functions for each of the two input nodes. The step size in our experiments was initialized to 0.01. This value was selected after experimenting with several initial values. The complete network configuration is shown in Figure 1.

3.2 Database Description

The database used for the system evaluation was composed both of some selected sentences from the well-known TIMIT database, and of an Italian database collected by us. The later contains recordings of four speakers (2 males and 2 females). The recording was made in a quite room, using a sampling rate of 16 kHz, in mono-channel at 16 bits per second. The sentences taken from TIMIT database are 6 (two dialectal sentences, two phonetically-compact sentences and two phonetically diverse sentences). A total of 22 sentences were used for all the experiments. The length of the recorded sentences was different in all the cases. In general the number of samples in the spoken sentences ranges from 36781 to 114800 samples. Each sentence was corrupted with each of the four kinds of noise (white noise, car noise, traffic noise and babble noise), giving a total of 88 noisy sentences.

3.3 Experimental Design

In our previous reported papers [15,16], the system was trained on all 88 noisy sentences in order to estimate $d(t)$ and then, the sentences were cleaned sub-

stracting the estimated $\widehat{d}(t)$ from the noisy sentences. For those experiments, we were using only two membership functions for each of the two inputs, which altogether generate a FIS of only 4 rules. Since the number of membership functions and the number of rules were small, the system had to be trained intensively to come up with a good performance. For this reason, in the mentioned previous works, we had to use 100 and 60 training epochs respectively. Due to the fact that the training was repeated for each of the 88 noisy sentences, the overall computational time was very high.

For the present work, in order to avoid the retraining time and to improve the system's performance, we decided to increase the number of membership functions for each input node, and after training only once the system, process all the noisy sentences over the FIS obtained. Following these goals, and after some tests, we opted for training with the longest sentence in our database, corrupted with the babble noise (which is a mixture of very different noises). On these grounds we follow the procedure described next.

We first take the longest clean speech sentence from the database, which is the primary input to the implemented system. A sample of noise of the same lenght is also collected. During the process, this noise sample is first delayed, and then distorted by means of the passage dynamics function described in the equation 3, whose output $d(t)$ is used to intercept the speech signal $x(t)$. In this way, we obtain the detected and measurable signal $y(t)$ that is expressed by:

$$y(t) = x(t) + d(t) \tag{9}$$

The number of inputs to ANFIS for the training process depends on the known order of the passage dynamics that the noise goes through. In this case, we applied a second order function, and for this reason we required only two inputs, which are $n(t)$ and $n(t-1)$, which come from a clean sample of babble noise collected aside. As target we used the detected signal $y(t)$. Looking to the desired output in this way, the original signal $x(t)$ is treated as an unwanted component of the measurable wave. The goal of our system is to estimate $d(t)$ through the ANFIS training process. After which, we can obtain an estimate $\widehat{x}(t)$ of the original signal by subtraction of the estimated noisy component $\widehat{d}(t)$. The stop criterion of the training process is based upon the error goal, set to 0.01, or the number of training epochs, which was set up to only 3.

After the learning is completed, the output of ANFIS is a fuzzy inference system (FIS), with all the parameters adapted to simulate the effects of the passage dynamics functions over the noise wave. The final FIS is first saved, and latter applied to get an estimated $\widehat{d}(t)$ of the distorted noise, which is expected to be as close as possible to $d(t)$. Then, the estimated $\widehat{d}(t)$ is subtracted from the detected signal $y(t)$, to obtain an estimation $\widehat{x}(t)$ of the original speech signal $x(t)$. The process, without new training, was applied to clean each of the 88 noisy sentences described above.

Table 1. Summary of Results

	Length	Δ SNR	MSE	AE
Avrg	73090	37.14 dB		
Min	36781	33.16 dB	$54.23e^{-6}$	$7.37e^{-3}$
Max	114800	38,45 dB	$59.54e^{-6}$	$7.47e^{-3}$

Table 2. Summary of Previous Results

	Length	Δ SNR	MSE	AE
Avrg	73090	36.21 dB		
Min	36781	20.51 dB	$11.0e^{-9}$	$6.9e^{-5}$
Max	114800	45.98 dB	$27.15e^{-5}$	$16.38e^{-3}$

4 Evaluation of Results

4.1 Quantitative Evaluation

To quantitatively evaluate the performance of the implemented system, we compared each original sentence $x(t)$ to the estimated sentence $\widehat{x}(t)$ using three different comparison criteria, namely; Signal to Noise Ratio (SNR), Mean Square Error (MSE), and Average Error (AE) [15,16]

Table 1 summarizes the statistical results obtained after processing the 88 different experiments with the present system, and Table 2 represents the previous results in [16]

Several observations have to be made on the results shown. First, the results in Table 1 were obtained after three training epochs, using 7 membership trapezoidal functions for each input variable, and using the same FIS, without new training, to evaluate and clean the 88 noisy sentences. The results in Table 2 were obtained training the system over 60 epochs, with 2 membership functions for each input variable. The training was done for each noisy sentence to be processed. The same second order *sinc* passage dynamics function was used in both cases. Taking these differences into account, we can point out that, although the more recent results show only a small SNR improvement over the previous ones, they can seldom be compared with reference to the computational time required to obtain each of them. Since the present system does not require new training, the processing of each noisy sentence was done almost in real time, while in the previous system the training and processing of each sentence took an average of 13 minutes.

In Figure 2 we show the initial Signal to Noise Ratio (SNR1) and the Signal to Noise Ratio after cleaning all 88 testing sentences. As can be seen, there is a substantial improvement in all the cases, regardless of the speaker, the language, or the noise used.

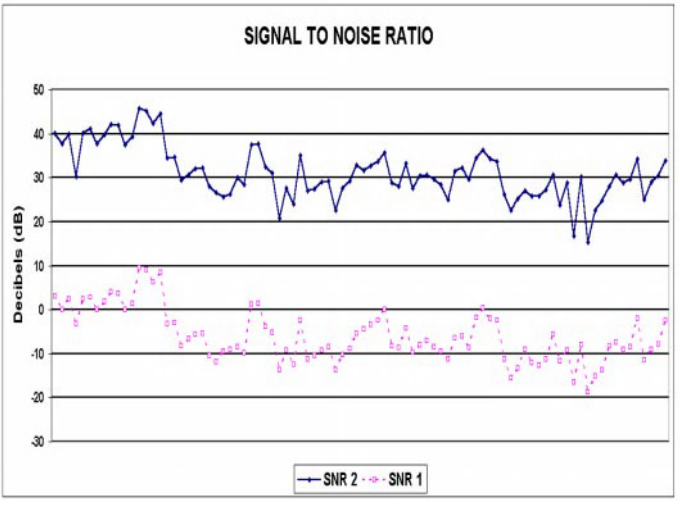


Fig. 2. Signal to Noise Ratio in the noisy sentences (SNR1), and the resulting Signal to Noise Ratio (SNR2) after processing the 88 sentences. Each point in the graph corresponds to one sentence.

4.2 Qualitative Evaluation

Besides collecting the quantitative information from the experiments, the results were at the same time qualitatively evaluated. This was done by the careful observation of the spectrograms and the waveforms of the clean input sentence, the noisy sentence, and the estimated sentence after being cleaned. The waveforms corresponding to the input noise, the distorted noise, and the output noise estimated by the system were also included for observation and analysis. Furthermore, given that our main objective is to obtain a clean and understandable speech signal, and that for further processing or recognition of the speech, quantities, spectra, or waves might mean nothing, we perform a perceptual evaluation over the estimated sentences. This evaluation consisted in listening to the cleaned sentences, trying to identify any loss of information during the cleaning process. In general, all the processed sentenced were very clear and the noise was almost unnoticeable.

The set of observations and perceptual tests performed, helped us to ratify the quality of the speech obtained after the noise canceling process. As an example of the several stages of the process observed, we include the following figures. Figure 3 includes, from top to bottom, the waveform of a sample of babble noise after being distorted by the passage dynamics function, as well as the waveform of the same noise as estimated by the system. Figure 4 shows, from top to bottom, the waveforms corresponding to one sentence corrupted with the babble noise, and to the same sentence after cleaning. And, the Figure 5 represents, from top to bottom, the spectrograms of another noisy sentence, corrupted with traffic noise, and the same sentence once cleaned.

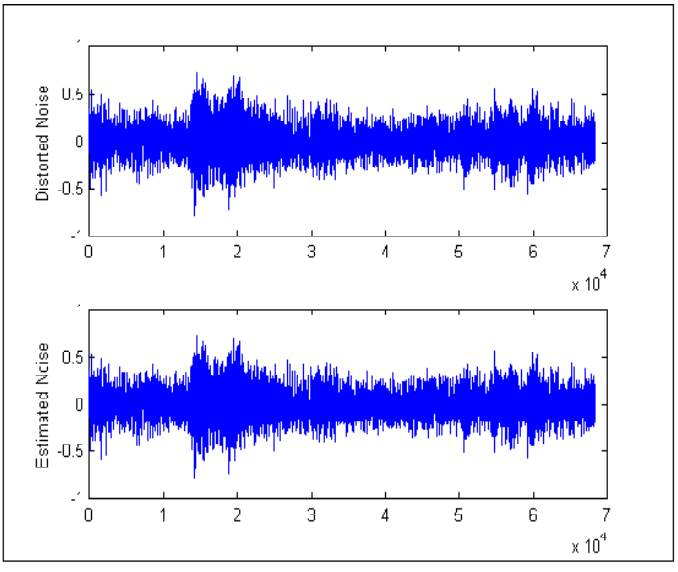


Fig. 3. Waveform of Babble noise after distorted by the passage dynamics function (top), and the waveform of the noise estimated by the system (bottom).

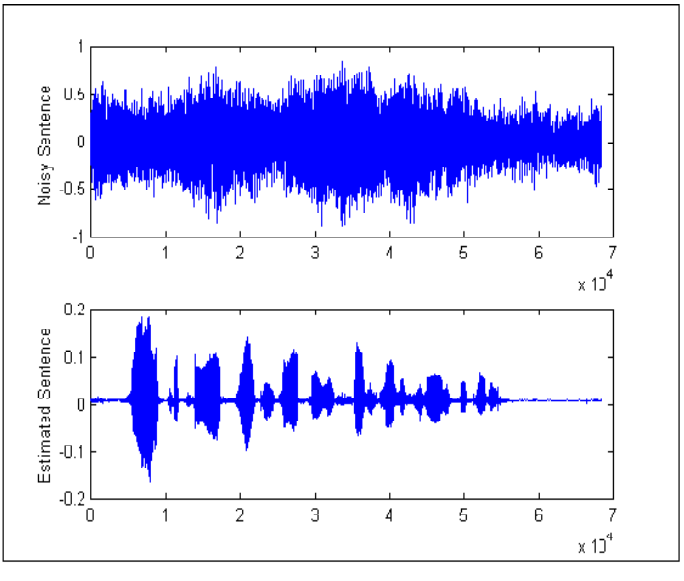


Fig. 4. Waveform of a noisy sentence corrupted with Babble noise (top), and the corresponding to the same sentence after canceling the noise (bottom.

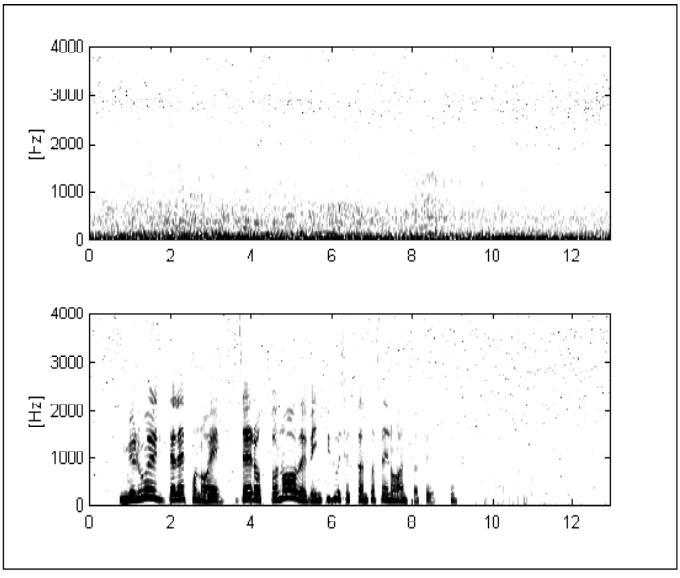


Fig. 5. Spectrogram of a noisy sentence corrupted with traffic noise (top), and the spectrogram of the same sentence after canceling the noise (bottom).

5 Concluding Remarks

The system configuration and the experimental approach reported in this paper, have yield a fast and efficient system, with the capability of canceling noise from speech in a computational time close to real. As mentioned above, the system was trained only once with the longest sentence in our database (114800 samples), which was corrupted with babble noise. The FIS obtained in this way allows the cancellation of any type of noise, from any sentence, coming from any speaker and regardless of the language. This performance can be explained by several considerations. In the first place, since the babble noise is a mixture of different noises, some characteristics of the other kinds of noise sources are learned during training. Second, given the adaptive nature of ANFIS, after training with enough number of samples, the resulting FIS is able to simulate the dynamical distortion added by the environment to the noise, allowing the remotion of any noise passing by the same medium. Third, the use of seven fuzzy membership functions gives a more detailed description of the input (the noise). In this case, although the number of rules increases with the number of membership functions, the system requires less training epochs to produce acceptable results. To obtain the results reported here, our system required only three training epochs. Moreover, since the training was made off-line, the computational time required for this task is not an significant factor in the further processing.

The drawback of this system is that it requires to have a clean sample of noise, both for training and for evaluation (which is a requirement for any adaptive

system for canceling noise). This fact limits its application to situations where the sample of noise can be obtained in parallel with the noisy speech. Further research should be done to overcome this limitation.

Acknowledgments. Dr. Ezin and Dr. Reyes-Garcia participation is supported by IIASS-Eduardo R. Caianiello.

References

1. Widrow, B., Glover, J., R., et al, "*Adaptive Noise Cancelling: Principles and Applications"*, in IEEE Proceedings, Vol. 63, No. 12, pp. 1692-1716, Dec. 1975.
2. Widrow, B., and Stearns, S. D., *Adaptive Signal Processing,* Prentice Hall, Englewood Cliffs, N.J., 1985.
3. Haykin Simon, *Adaptive Filter Theory,* Prentice Hall International, Upper Saddle River, New Jersey, 1996.
4. Avendano, C., and Hermansky, H, *"On the Properties of Temporal Processing for Speech in Adverse Environments",* Proceedings of 1997 Workshop on Applications of Signal Processing to Audio and Acoustics, Mohonk Mountain House, New Paltz, New York, 1997.
5. Hermansky, H, *"Analysis in Automatic Recognition of Speech",* in Speech Processing, Recognition, and Artificial Neural Networks, Edited by Chollet, G., et al, Springer-Verlag, London, 1999.
6. Avendano, C., *Temporal Processing of Speech in a Time Feature Space,* Ph.D. Dissertation, Oregon Graduate Institute of Science and Technology, April 1997.
7. Wan, E., and Nelson, A., "*Neural Dual Extended Kalman Filtering: Applications in Speech Enhancement and Monaural Blind Signal Separation*" in Neural Networks for Signal Processing VII: Proceedings of the 1997 IEEE Workshop, Ed.. Principe, Morgan, Giles, Wilson, 1997.
8. Wan, E., and Nelson, A., "*Networks for Speech Enhancement*" in Handbook of Neural Networks for Speech Processing, Edited by Shingeru Katagiri, Artech House, Boston, 1998.
9. Jang, J.-S. R, Sun, C.-T., and Mizutani, E., *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence,* Prentice Hall International, Upper Saddle River, New Jersey, 1997.
10. Jang, J.-S. R., *"ANFIS: Adaptive Network-based Fuzzy Inference System",* in IEEE Transactions on Systems, Man, and Cybernetics, 23(03):665-685, May 1993.
11. Jang, J.-S. R, *Neuro-Fuzzy Modeling: Architectures, Analyses and Applications,* Ph.D. Dissertation, University of California, Berkeley, July 1992.
12. Jang, J.-S. R, Sun, C.-T., *"Neuro-Fuzzy Modeling and Control",* in The Proceedings of the IEEE, vol. 83, pp. 378-405, March 1995.
13. Lin, Chin-Teng, and George Lee, C. S.., *Neural Fuzzy System: A Neuro-Fuzzy Synergism to Intelligent Systems,* Prentice Hall, Upper Saddle River, NJ, 1996.
14. Speaks, Charles E., *Introduction to Sound: Acoustics for the Hearing and Speech Science,* Singular Publishing Group Inc., San Diego CA, 1997.
15. Esposito, A., Ezin, E. C., Reyes-Garcia, C. A.., *"A Nonlinear Adaptive System to Cancel Noise from Speech",* in WILF'99 Proceedings, Genova, Italy, June 1999.
16. Esposito, A., Ezin, E. C., Reyes-Garcia, C. A.., *"Speech Noise Cancellation on a Neuro-Fuzzy System: Experimental Results",* in IEEE's WISP'99 Proceedings, Budapest, Hungary, pp. 342-347, Sept. 1999.

Verification of Correct Pronunciation of Mexican Spanish Using Speech Technology *

Ingrid Kirschning and Nancy Aguas

Tlatoa Speech Processing Group, ICT, CENTIA**, Universidad de las Américas-Puebla. Sta. Catarina Mártir s/n, 72820 Cholula, Pue., México.

Abstract. This paper presents a new method for the verification of the correct pronunciation of spoken words. This process is based on speech recognition technology. It can be particularly useful when applied to the field of SLA (Second Language Acquisition) in learning environments or Computer-Aided Language Learning (CALL) systems, where the students can practice their pronunciation skills. This method uses an artificial neural network plus a specific grammar for each utterance to compare the text of the expected utterance with the sequence of phonemes recognized in the speech input, in order to detect the pronunciation errors.

1 Introduction

Due to the latest developments in computer technology regarding processing speed, audio devices and storage, the quickly evolving speech technology can now be used in many different applications, able to run on personal computers. This fact makes the newest speech technologies accessible to anyone who can use a computer. Among those applications that seem most suitable for the integration of speech interfaces are the language learning systems. There is a large variety of speech interfaces integrated into computer aided language courses.

There are several commercial products already on the market but none provides an explicit feedback about the pronunciation errors. They usually provide an overall score for the complete utterance. Others display the soundwave of the "correctly" pronounced word and the soundwave of the word uttered by the user. The user should then be able to detect his/her own errors by comparing both soundwaves. None of the products found, showed an efficient way to find out exactly why and where an utterance is not accurately pronounced.

This paper presents a method that enables an automatic speech recognition system to perform an explicit analysis of the correct pronunciation of words and/or phrases. The prototype shown here is part of an undergraduate thesis of one of the members of the TLATOA Speech Research Group at the UDLA (Universidad de las Américas-Puebla. It is also a part of a larger project of TLATOA concerning the development of a second language learning environment applying speech technology.

* Research funded by CONACyT, project No. I28247-A

** Research Center for Information and Automation Technologies.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 493–502, 2000.

The following sections introduce the motivation for this work and some examples of software products created for second language learning. Then it presents a new method using neural network based speech recognition to check pronunciation.

The verification method is implemented in a prototype, where a simple interface has been designed to test the recognizer's ability to spot right and wrong pronunciations. The results obtained so far are presented in the last section followed by the conclusions and further work for the near future.

2 Computer Asisted Language Learning (CALL)

Computer Assisted Instruction (CAI) is a means through which computers aid users in the process of learning. Over the past 40 years, there has been an exponential growth in the use of computers assisting in instruction. During this rapid technological revolution, CAI has become more refined as a result of the paradigmatic shifts from behaviorist thinking to cognitivism to constructivism. Computers now serve as a sophisticated medium through which instruction can be delivered.

Symbolic AI proposed interesting schemes in ICAI, but the use of neural networks has been proposed very few times. In general the knowledge for CAI is rather represented explicitly, therefore neural network architectures have rarely been used in this area [1].

CAI gives way to the discovery mode of teaching and learning whereby the student is involved much more freely in self-directed intellectual exploration of the knowledge or the skill domain incorporated in the system. It serves as a tool through which learning can occur. In the last years more and more CAI systems have incorporated multimodal interfaces, i.e. sound, images, video and interactive programs to support various applications. More recently, the incorporation of speech interfaces into the area of CALL or ICALL (Intelligent Computer Aided Language Learning) creates new possibilities for the development of learning tools [2].

3 Speech Technology and Education

In the various fields of language education, different groups have developed interesting and useful applications. For example, there is a software product called Pronunciation Power [3], which presents to the users a collection of tools that permit them to learn about the pronunciation of English language. It presents the written form of the sounds, lets the users record their own speech and compare every sample to a "correct" version of the soundwave.

The apparent disadvantage of this method is that when there is a pronunciation error this is not stated explicitly. It requires practice to analyze a soundwave and determine if the variation should be considered an error. It might be relatively easy for an adult user, but it might be tedious. Additionally, it is not known under which criteria the "correct" waveform was chosen. A correct utterance can

vary in its appearance enough due to the natural differences in the human vocal tract to make the user believe that something might be wrong.

Another interesting example is from *Language Connect* [4], which uses IBM's ViaVoice. The software "listens" to each word and phrase, even complex sentences, then responds whether the user would be understood by a native speaker, also giving a general score for the pronunciation of the whole word or phrase. The speech recognition software is very powerful and has a good accuracy; however, this score the student receives also doesn't state explicitly which part of the utterance was not correctly pronounced, nor how it can be corrected.

In another application field, the Center for Spoken Language (CSLU) [5] has been collaborating with educators at the Tucker Maxon Oral School [6] in a joint effort focused on speech training with profoundly deaf children. They have developed a Toolkit that incorporates speech recognition and synthesis facilities, as well as an animated conversational agent, called Baldi [5] [7] [8]. The agent is represented by an animated 3D face that produces visual speech: facial expressions, eyebrows, lip, jaw and tongue movements during speech production using a synthesized voice [9]. The children can play with the speech interface where a lesson presents different questions and a correct answer permits them to continue (see figure 1).

Apart from the software mentioned above there exist a large variety of tools, programs and games that intend to help people to learn a language. Most of these tools are quite impressive and they adapt to different needs of the users.

TLATOA is collaborating since 1997 with CSLU in the construction of the Mexican Spanish version of the CSLU Toolkit [10]. At the UDLA, as in many other institutions, foreigners spend their summer in language courses to learn Spanish. A large percentage of these students come from the United States.

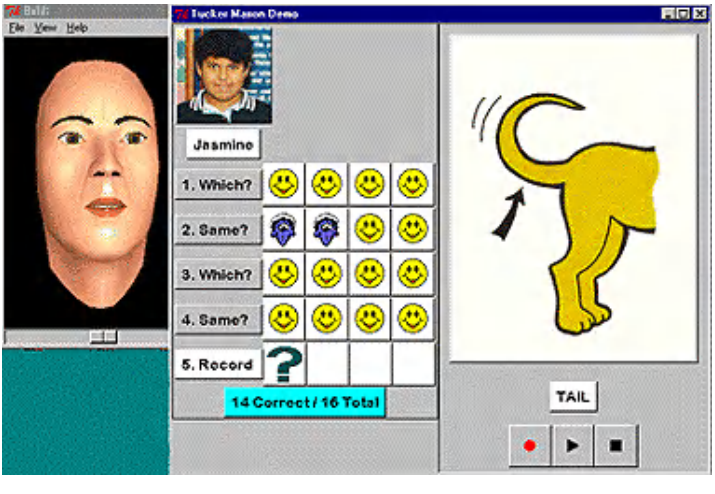


Fig. 1. An example of the animated agent and a language training lesson [5] [7] [8]

The ever-increasing demand of Spanish courses has prompted us to think of other means to provide suitable learning environments for the students. Thus a project for the development of a Mexican Spanish learning environment has been initiated.

As a part of this project that is still in an early stage, we focused on the problem of verifying the correct pronunciation of a word or phrase. Since language learning is about to be able, not only to write it, but also to speak, the training of a correct pronunciation is obviously important.

4 Verifying Correct Pronunciation

4.1 System Architecture

The main objective of this work is to test how far a neural is capable of detecting mispronunciations in a stream of speech. In a first step the recognizer has to recover the exact sequence of phonemes as the user pronounced them, without neither correcting any nor fitting them to the correct word model. A normal speech recognition system processes the speech input and produces as an output one of the words or phrases of the vocabulary with the highest similarity score. For this purpose the speech recognition system uses a vocabulary and a grammar to try to map the soundwave to one of the options in the grammar, rejecting results with low recognition confidence (below a determined threshold). These systems usually try to overcome noise and some errors to find the closest words - although not always corresponding to the input steam. The more robust a system is, the better, because it is able to produce the correct output most of the times.

Applying this to a language training system, a student can be asked to try and pronounce a set of words according to the level of difficulty of the lesson. A speech recognition system can easily be used to try to recognize the student's speech and give a value of certainty of the recognition. We believe that it would be extremely useful to be able to tell the student also which part of the word was correctly said and where an error occurred, strong enough to be noted. This would help to tailor the lesson to the particular weaknesses of each learner, and support them in the practice of those parts they wish to improve.

However, in our case we wouldn't want the system to reject an utterance as "wrong", just because it is not a valid word in the vocabulary. We neither want it to find something that sounds similar (its best match). The task is to check every utterance phoneme by phoneme and compare what was said and what should have been said. The part of what should have been said is easy to know, since the system manages the lesson being practiced. The problem of detecting what was said needs a recognizer that was trained to detect all kinds of sounds (phonetic contexts) and transcribes them correctly without searching the full words or phrases out of a vocabulary. If the utterance contains pronunciation errors, most probably the word won't be in the vocabulary, and we don't want it to be adapted to the existing vocabulary nor rejected as "out of the vocabulary".

4.2 Using the Speech Recognition Approach

In a first attempt we experimented with a neural network trained for Mexican Spanish speech recognition and defined a vocabulary consisting only of the letters of the alphabet and their phonetic transcriptions. The grammar allows any combination of these letters. For this approach the recognizer had to be extremely accurate. It should preferably be speaker dependent and tested in a noise free environment, avoiding even breathing noise into the microphone. Otherwise it would insert a large quantity of garbage between the correctly recognized phonemes.

A way to solve this would be to restrict the grammar to only the words necessary for the application, but this would cause the system to reject the wrongly pronounced words. Another option would be that the vocabulary includes an extensive list with every word and all it's possible mispronunciations. This increases the search space too much, causing again an increased error rate, particularly a high number of elimination errors.

To solve the problem we adopted some ideas from a technique we use for labeling speech corpora automatically. This technique is known as forced alignment [11] [12] [13] [14]. This technique aligns a soundwave to its textual transcription, marking the beginning and ending of each word and/or phoneme within the soundwave. In order to do so, forced alignment requires a previously trained neural network (or another recognizer), the soundfile, it's textual transcription and a vocabulary. It then takes the transcription, creates a specific grammar fitted only to the text being aligned, and then processes the sound, frame by frame, classifying and mapping them to the small specific grammar, annotating the position in milliseconds where each phoneme starts and ends.

The set of phonemes used in Mexican Spanish is practically a subset of the phonemes used in English; therefore, a recognizer trained for Mexican Spanish will not detect phonemes outside this phoneme set. Thus, to be able to detect mispronounced phonemes the recognizer has to be trained all the possible sounds and their contexts.

Using the approach mentioned before, the verification process has to generate first the specific grammar for the expected utterance. This is based on the vocabulary, which contains all the words that can be practiced together with their phonetic transcriptions. However, these transcriptions also need to include all of the other possible pronunciations of every phoneme in the word, correct and incorrect. For example:

The system can take the word that is expected to be uttered by the learner. For example if the user is expected to say "ABEJA" the system first creates a grammar using a transcription chart as in table 1 for the possible pronunciation of the word including the *incorrect* ones. This specific grammar looks like the following example:

ABEJA = {{ a | ay | e | ey | uh}{ bc b | B }{ e | ey | i }{ x | h }{ a | ay | e | ey | uh} };

In the next step the neural network analyzes the input speech frame by frame trying to mach them to the specific grammar, recording the best matching

Table 1. Example of the transcriptions of some letters (correct and incorrect from the Spanish Language point of view)

Letter	Phonetic Transcription
a	{ a | ay | e | ey | uh} ;
b	{ bc b | B } ;
e	{ e | ey | i } ;
j	{ x | h } ;

sequence. If the user mispronounced the 'j' as an 'h' the network should output the sequence {a bc b e h a}. This result can then be used to compare it to the correct one, which is {a bc b e x a}. Then the system can detect if they differ and pinpoint exactly where they differ.

5 Training and Testing of the Prototype

With the above-mentioned method a prototype was implemented, programming the interface in Jacl [16] that invokes the recognizer and performs the pronunciation check. The reason for using Jacl, a not yet very robust version of Java, able to call functions written in Tcl, was that it provides the advantages of Java to program visual interfaces easily. It also allows the interaction with the scripts of the CSLU Toolkit, which are written in Tcl and C.

5.1 Verification Tool

The neural network for this experiment was trained with the samples of 400 speakers (8 MHz) plus samples from another corpus of 10 speakers containing the vowels, diphthongs and consonants in English. It is a feedforward-, fully connected neural network with 130 input units, 200 hidden and 366 output units. The network was trained as a general purpose-, context dependent recognizer, using standard backpropagation. It reached an accuracy of 95% after the 30th iteration in the training phase. The vocabulary for this experiment consisted in phonemes only and a grammar that permits any sequence of these phonemes with their various pronunciations. For example, the letter 'o' can have the pronunciation of the 'o' in Spanish or the 'ow' in English (as in "only"). This is expressed in the vocabulary as: o { o | ow } ;

Like this example all the other phonemes were expressed in terms of their possible pronunciations in order to have the recognizer pick the correct one. The main difficulty is to determine the possible pronunciations a user could give each letter, in order to be able to recognize them.

It has been found that the most common pronunciation mistakes a native English-speaker makes when learning Spanish are those sounds that are pronounced in English with an aspiration, like 'p', 't', 'k', 'b', 'd', and 'g'. Also the vowels and the consonants like 'r', 's', 'z', 'y' and the Spanish 'll', are difficult at first, specially when they don't exist in English, like the 'ñ' [15].

A separate database contains a list of the all the words in Mexican Spanish the student can train with, their translation into English, and its correct pronunciation. It also includes a short text giving a recommendation on the correct pronunciation of each letter in the word. Additionally we recorded a sample pronunciation of each word and an image that illustrates the concept of the word; both identified by the word they are related to.

The system allows the user to navigate through menus to select the topic and the word for practice. It then allows the student to record his/her speech and when he/she chooses, it will verify the pronunciation. This means that the recognizer classifies the sequence of phonemes in the uttered word, and compares the result to the word the user was supposed to say, marking the differences.

Table 2. An axample of the transcriptions and pronunciation mistakes

Expected Word	Transcription	User said
ABEJA	a bc b e x a	a bc b **ey** x a
CUATRO	kc k w a tc t r o	kc k w a tc t r **ow**

In the above example, the first column contains what the user sees on screen and is going to pronounce. The transcription in the second column is the internal representation the recognizer uses as the correct version (there can be more than one). The third column is an example of what a user could have said with some of the common pronunciation mistakes. The correct phonetic transcription and the one obtained from the users utterance are compared phoneme by phoneme trying to detect the existence of an error. The result is stored in a file. It should be noted here that the plosives, like p, t, k, or b, d and g, which are divided into two parts, the closure and then the phoneme, are treated as one phoneme when comparing the pronunciation results, i.e. as a 'k' and not as 'kc k'.

The types of errors that can be encountered are insertion, elimination and substitution errors. Insertion errors occur when the user says more than he should or when an external noise causes the recognizer to insert more information. An elimination error means that a phoneme was skipped by the same reasons. At this point the system handles only the substitution and simple insertion and elimination errors by checking the recognition result and the correct transcription phoneme by phoneme. When a mismatch is encountered it looks forward one phoneme to see if it can recover the sequence. When a point is reached where nothing matches the system asks for the student to try again from the beginning.

To test the method we focused on the development of a system able to help students whose native language is American English in their Mexican Spanish language learning course. The domain for the vocabulary was chosen to be animal names, food and digits.

5.2 Interface Design

This section shows the interface designed to test the verification method. Figure 2 shows two examples of the screens of the prototype. When the user enters the

first screen he/she can choose among different topics. For every topic there will appear a list of words that the database contains for practice. When a certain word is chosen the system plays it's pronunciation example and displays the image illustrating the words meaning, at the same time showing it's translation in a separate field. The lower text field presents a recommendation for the pronunciation of the selected word. The left side of figure 2 shows the screen when a user has chosen the topic "ANIMALES" and then the word "ABEJA".

From the "Options" menu above, the user can choose to record a speech sample, where he/she will be prompted with a set of buttons to control when to start and stop the recording, as well as play it.

Using the same menu the user can choose the option of verifying the recorded sample. At that point the system presents the screen shown on the right side of figure 2, where the text field at the bottom now contains the verification results.

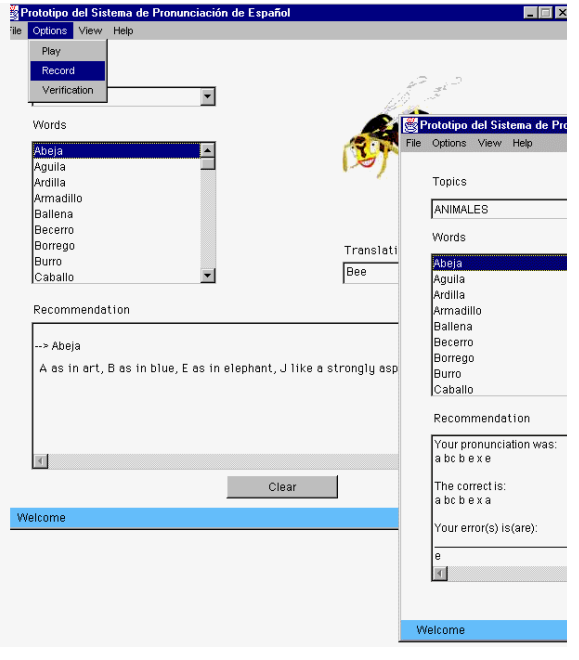


Fig. 2. The prototype's screens, showing the information the system can provide to the user

5.3 First Results

The previously explained interface calls the necessary processes to run the neural net-based verification and then analyzes the outcome. At this point the system is able to cope with simple errors, mainly one-character substitution, insertion and elimination errors.

To test the system a group of five foreign students were asked to record each a set of 12 sequences of words. First, these recordings were evaluated by 15 native Spanish speakers, who marked every error at the phoneme-level. Next the system was run to analyze these recordings as well and the differences recorded in a log.

The speech samples recorded by the students were sequences of words belonging to various topics (food, animals, etc.). The recordings contained a total of 256 phonemes each. The human evaluators detected a total of 86 errors (6.72%), whereas the system found only 62 errors. Not all the errors marked by the system matched those of the evaluators. The results of the automatic verification matched only a 98.28% of the human evaluations. There was a 5.16% of mispronounced phonemes not detected by the system. Further 3.44% of the phonemes were marked by the system as mispronounced but not by the human evaluator [17].

6 Conclusions

This paper presented a new approach to verify the pronunciation of spoken language. It uses a neural network based speech recognizer and a vocabulary of the letters of the alphabet, which include all the possible correct and incorrect pronunciations of every letter. The neural network was trained as a general purpose-, context dependent recognizer, with samples from more than 400 speakers and it has an error rate of 5% during the training phase.

So far the results are satisfactory, which means that when intentionally mispronouncing a word in the vocabulary the system detects it. The difference between th human and the automatic evaluation should be treated with care, as this kind of evaluation is highly subjective. However, we discovered that the neural network was not trained with enough erroneous phonemes in all the possible contexts of Mexican Spanish. Thus it is necessary to find more samples and re-train the network.

The evaluation done by native Spanish speakers and the automatic pronunciation check matched a 98.28% of the cases (verified phonemes). These results obtained so far are promising but still need much more testing with a larger number of foreign students and a larger vocabulary.

Some phonemes that were found to be particularly difficult for the system to verify were the /x/, the pronunciation of the letter 'j', and /dZ/ the reading of 'll'. Additionally, not all the possible contexts of the correct and incorrect phonemes were represented in the training set for the neural network. This forced us to eliminate a large portion of the test set.

The presented method offers a way to check and pinpoint exactly the mistakes in pronunciation of a word or phrase. The recognizer is still restricted to fit the recognition to a grammar and a predefined vocabulary, but the method inspired by the one used for forced alignment eliminated the insertion errors in the output. The actual vocabulary of the prototype is small and should be tailored better to the real needs of the Mexican Spanish language learner. However, this first part of the development of a Mexican Spanish language learning environment that uses the CSLU Toolkit shows an interesting potential for automatic pronunciation checking.

References

1. Ayala, G.: Personal communication (1999)
2. Bull, S.: Student Modelling for Second Language Acquisition,. Computers and Education (1994)
3. Pronunciation Power, http://www.englishlearning.com/
4. Language Connect, http://shop.languageconnect.com/
5. Center for Spoken Language Understanding, http://cslu.cse.ogi.edu/tm/
6. Tucker Maxon Oral School, http://www.oraldeafed.org/schools/tmos/index.html
7. Cole, R., Carmell, T., Connors, P., Macon, M., Wouters, J., de Villiers, J., Tarachow, A., Massaro, D., Cohen, M., Beskow, J., Yang, J., Meier, U., Waibel, A., Stone, P., Fortier, G., Davis, A., Soland, C.: Intelligent Animated Agents for Interactive Language Training. STILL:ESCA Workshop on Speech Technology in Language Learning, Stockholm Sweden, (May 1998)
8. Cole, R., Massaro, D., de Villiers, J., Rundle, B., Shobaki, K., Wouters, J., Cohen, M., Beskow, J., Stone, P., Connors, P., Tarachow, A., Solcher, D.:New Tools for Iinteractive speech and language training: Using animated conversational agents in the classroom of profoundly deaf children. Proc.ESCA-Matisse ESCA/Socrates Workshop on Method and Tool Innovation for Speech Science Education, London, UK. (April 1999)
9. Black, A., Taylor, P.:Festival Speech Synthesis System:System documentation (1.1.1.). Human communication Research Centre Technical Report HCRC/TR-83, Edinburgh, (1997)
10. Serridge, B., Cole, R., Barbosa, A., Munive, N., Vargas, A.:Creating a Mexican Spanish version of the CSLU Toolkit. Proc. of the International Conference on Spoken Language Processing, Sydney, Australia (November 1998)
11. Young, S., Odell, J., Ollason, D., Valtchev, V., Woodland, P.:HTK Book, (1997) http://ceres.ugr.es/HTKBook/HTKBook.html
12. Olivier, A.:Evaluación de métodos de determinación automática de una transcripción fonética. Undergraduate thesis, Dept. of Computer Systems Engineering, Universidad de las Américas, Puebla, México (May 1999)
13. Olivier, A., Kirschning, I.:Evaluación de métodos de determinación automática de una transcripción fonética. Proc. of the II. Encuentro Nacional de Computación 1999, ENC'99, Pachuca, Hidalgo, México (September 1999)
14. Hosom, J.P.: Accurate Determination of Phonetic Boundaries Using Acoustic-Phonetic Infomation in Forced Alignment. Thesis Proposal, CSLU, Oregon Graduate Institute (August 1998)
15. Dalbor, J.B.: Spanish Pronunciation:Theory and Practice. Pennsylvania State University, Holt, Rinehart and Wiston, Inc., (1969)
16. http://scriptics.com/products
17. Aguas, N.:Verificación de pronunciación para un ambiente de aprendizaje basado en tecnología de reconocimiento de voz. Undergraduate thesis, Dept. of Computer Systems Engineering, Universidad de las Américas, Puebla, México (December 1999)

Explaining Abstract Data Types with Sentential and Diagrammatic Elements

Agustin A. Araya and Jessica H. Chu

Dept. of Mathematics and Computer Science
California State University at San Jose
San Jose, CA 95192, USA

Abstract. Although much work has been done on Intelligent Tutoring Systems that support the learning of programming skills, little attention has been paid to supporting the understanding of programs. We discuss the issues involved in developing a tool for a learning environment in which novice programmers can explore in detail an important class of programs implementing Abstract Data Types (ADTs). We start by developing a conceptual framework for explaining ADTs. Next, we describe an interactive learning situation and discuss the design principles that it embodies. After presenting an example of student/system interaction, we describe in detail the explanation process underlying it. We end with a discussion of the main issues.

1 Introduction

Most of the exploratory applications of Artificial Intelligence that support teaching and learning of introductory topics in Computer Science have been directed towards supporting the learning of programming [3]. Although programming skills are crucial in computer science, the ability to understand existing programs is equally important, in particular, to gain a detailed and precise understanding of Abstract Data Types (ADTs), their use, and their implementation.

ADTs are a central topic in introductory computer science courses. Even if students are able to implement simple ADTs and to use them as components in larger programs, many students do not develop an understanding of how specific ADTs work in detail, understanding that would allow them to explain their functioning and to transform them to satisfy new requirements. Textbooks in introductory data structure topics offer textual explanations of specific implementations of ADTs, explanations which are typically accompanied by diagrams illustrating the data structures and their basic operations. But textbooks, because of their "non-interactive" character, do not allow students to probe with questions whose answers are not given in the provided explanations. As a result, students are left with gaps in their knowledge of ADTs, gaps which may have negative consequences on how they use them.

In the last decade, the notion of software visualization has been extensively developed as a way to provide visual "explanations" of how algorithms work [6], [13]. An important limitation of this approach is that the connections between the code imple-

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 503—514, 2000.

menting the algorithm and the graphical events shown are not always made explicit. As a consequence, in spite of seeing a visualization questions students may have about particular statements in the code may remain unanswered.

In this work, we discuss the issues involved in developing a particular tool of a learning environment with which beginning students can explore in detail an important class of programs implementing ADTs, and we present a prototype implementation of it. We start by developing a framework for explaining ADTs. Next, we describe a learning situation and discuss the design principles embodied in it. After presenting an example of a student/system interaction we describe in detail the explanation process supporting it. We end with a discussion of the main issues.

2 Explaining ADTs

Intelligent Tutoring Systems (ITSs) support a learning situation in which a student is supervised by a system while solving a given problem. For the most part, ITSs in the area of computer science education support the task of programming [e.g., 8]. But the ITS approach has significant limitations [1]. By adopting a tutoring model, an ITS tends to put the student in a passive position while it is being monitored by the system. In addition, it has proven difficult to construct reliable models of student knowledge, without which an ITS cannot properly function.

On the other hand, in the so-called Interactive Learning Environments (ILEs) students learn by exploration, thus taking a more active role [4], [16], [17]. Andriessen and Sandberg [1] identify three pedagogical scenarios in which learning and instruction can take place, namely, transmission, studio, and negotiation. In the Studio scenario, the kind of scenario supported by ILEs, students assume increasing responsibility for learning, proceeding at their own pace, and following their own paths. It is this kind of scenario that the work presented here assumes.

Let us now examine the concept of an ADT to identify what is involved in understanding specific cases of it. Most ADTs of interest turn out to be "containers" storing collection of elements, and can be conceptualized as follows. A container ADT is a computational *representation* of a process consisting of a container and a set of subprocesses that operate upon it. As such, the various aspects of an ADT represent corresponding aspects of the associated container process. In addition, an ADT can be *implemented* in various ways. Any particular ADT implementation uses one or more lower level data structures and their elements. Finally, an ADT is used in the context of a larger program. In this respect, the distinction between interface and implementation becomes relevant.

To understand an ADT, then, implies to understand what it represents, how is it implemented, and how it can be used. Although these three kinds of understandings are important, in our current work we have concentrated on the first two cases, leaving the use of ADTs for future work. Fig. 1 depicts the representation and implementation relationships in which an ADT is involved. Because "container processes" are "common-sense" processes that we encounter in daily life, we can assume that to a large extent they are well understood by students. We can also assume that students at-

tempting to learn ADTs are already familiar with the lower level data structures involved in their implementation.

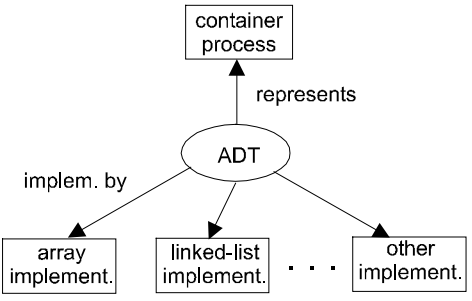


Fig. 1. ADT, Container Process and Implementations

In this context, then, understanding an ADT means to understand how the various parts of it represent various aspects of the container process, and how these parts are implemented in terms of lower level data structures. A particular way in which this can be achieved is by means of a question-answering process, that is, by asking the right kinds of questions about ADTs and by receiving appropriate answers to them, adequately explained. In the case of ADTs, explanations that make explicit the various kinds of relationships mentioned above should contribute to develop such an understanding.

An issue that needs to be addressed is the language in which explanations are to be expressed. In a classical work, Larkin and Simon [9] studied the relative advantages of diagrammatic representations versus logical representations. In diagrammatic representations of problems, in particular, of simple physics problems, the visual character of the representation makes it easier to draw certain kinds of inferences because the relevant facts for making them appear in close physical proximity. This is not necessarily the case with logical representations, in which making inferences typically involves a search process. Working along these lines, Cheng [5] developed the notion of law encoding diagrams which are a particular kind of diagrammatic representation that encodes "laws," such as physics laws, in terms of geometric and topological constraints. This kind of diagram allows for quantitative reasoning as well as for conceptual problem solving.

While diagrammatic representations have these useful characteristics, explanations in terms of natural language - which we will call "sentential explanations" - are also very important. First, they are the most commonly used form of explanation, and we are particularly adept at understanding them and, second, a sentential explanation can be extended with explanations of any one of its own terms, in a recursive way, characteristic which cannot be easily achieved with diagrammatic explanations.

In the generation of sentential explanations two very different approaches can be followed. A discourse-based approach makes explicit use of syntactic, semantic, and intentional components (e.g., [11]), while a template-based approach does away with syntactic processing concentrating on semantic components (e.g., [15]). Although the

first approach may produce explanations which are better attuned to user's intentions, it can be considerably more complex, bringing with itself more problems than it solves. A variety of semantic components have been used to generate explanations, including task models and qualitative domain models [12], [14]. Finally, we should mention that other approaches that have been used to provide explanations in the context of tutoring systems are based on hypertext [7], [10].

3 The Interactive Situation and Its Rationale

We now propose an interactive situation to be supported by the learning environment. Then, we examine it critically to justify the particular decisions embodied in it.

3.1 Learning Situation

Let us assume that the students engaging in the learning situation are novices who have a basic understanding of programming, are familiar with an object-oriented language, and are beginning to learn ADTs. Students are supposed to have already a general understanding of how a particular ADT works and use the interactive environment to obtain a *detailed* understanding of a specific implementation of it.

A variety of ADTs are available in the learning environment, from the simple stack and queue processes to the more advanced tree-based ADTs. For each ADT there is one or more implementation of it using different primitive data structures. Once the student selects for study a particular ADT and a particular implementation, the environment displays it in a Code Viewer window. For a simple ADT, this will consist of one or two class definitions and a set of basic functions that represent the operations of the ADT. As the student examines the code there may be parts of it that are only partially understood or not understood at all. By selecting a part of the code, selection which can be performed at various levels of granularity, the student signals the system that such part needs to be explained.

At this point, the system reacts by determining the kinds of questions that are meaningful to ask about the selected code. Any piece of code has a variety of meanings, which depend on the point of view from which it is being considered and on the particular context from which it is being examined. Each of the encompassing statements in which the selected code is embedded constitutes a possible context for asking a question about it.

Once the possible questions are determined, they are presented to the student. If the question the student had in mind does not closely match any of the presented options, the student still has the chance of selecting a question that may provide a partial answer to the original question. If the student had no specific question in mind but simply did not understand the code, the given questions constitute a space of possibilities the student can explore.

Finally, the system proceeds to handle the question selected by the student, and this is performed in two main steps. First, an "answer" to the question is determined, which is expressed in terms of an internal representation. Second, from this answer a

detailed explanation containing sentential and diagrammatic elements appropriately interleaved is elaborated and presented step by step to the student.

This is the basic functionality proposed for this explanation tool. It should be clear that this tool should be part of a more encompassing learning environment, that would include support not just for understanding how ADTs work but for actually writing programs that would use or modify ADTs.

3.2 Rationale

Decision 1: To let the system determine the possible questions that can be asked, rather than having the student formulate them.

We performed an informal study with students taking an introductory data structures course, in which we gave them a specific implementation of an ADT and asked them to report the questions they had about it. Several kinds of questions were identified: language questions (about C++), "meaning" questions (e.g., meaning and purpose of variables, statements, and functions), what-if questions (including possible effects of deleting or swapping statements in a function), design questions (about implementation decisions), and miscellaneous questions (e.g., comparison questions). According to the study, the first three kinds of questions constitute a significant fraction of the total. In addition, some of the what-if and miscellaneous questions can be reduced to meaning questions.

If we leave out questions on language, which are about a related but different topic, it turns out that meaning questions are the most significant. What this suggests, is that a set of predetermined meaning questions can have a good coverage of the kinds of questions students tend to ask. A similar situation in which the system offers multiple options to the student has been discussed in [2]. On the other hand, allowing the student to formulate a question creates all sort of potential difficulties including misinterpretation of the question and dealing with a question totally outside the scope of the system.

Decision 2: To circumscribe the domain of programs to be explained to a set of ADTs which behave as containers of objects.

The first reason for this decision is that ADTs are a central topic in introductory computer science courses. The second, is that they represent a cohesive body of knowledge consisting of a family of container ADTs bearing relationships to each other. This cohesiveness is important because it suggests the possibility of organizing the required knowledge base around the concept of container and the processes that manipulate it. Explanations, then, will be given explicitly or implicitly with reference to this basic metaphor of ADTs as containers.

Decision 3: To provide explanations consisting of sentential plus diagrammatic elements.

Sentential elements - that is, short pieces of text - and diagrammatic elements complement each other very well, in the sense that they compensate for each other's limitations. Sentential elements can be very explicit, but precisely, by being less condensed, it may be more cumbersome to make inferences with them. On the other hand, by presenting information in very condensed ways, diagrams facilitate the making of certain kinds of inferences. If the diagrams are based on simple, well- known visual metaphors, users are able to transfer their prior knowledge of the metaphor to the new situation they are trying to understand. But, if for any reason the user fails to make this transference, the visual explanation may not be fully understood, hence the need for complementary sentential explanations.

The diagrams to be used are simple, easy to understand visualizations of a container and basic data structures, including simple annotations.

4 A Detailed Example

We have implemented a prototype system embodying the learning situation previously described. Before examining how the explanation process is performed, let us consider an example student/system interaction. Assume the student has selected for study the Queue ADT implemented in terms of a circular array, and is now examining the "insert" function shown below.

```
void Queue::insert(const int entry)
{ if (size() < CAPACITY)
    {rear = nextIndex(rear);
     data[rear] = entry;
     count++;
    }
  else
    Error("queue overflow");
}
```

Suppose the student selects for explanation the function call "size()" appearing in the condition of the "if" statement. The questions that can be asked involving this particular function call are given below. Next to each item we have provided a brief description of a corresponding answer, answer which does not yet constitute the explanation that will be presented to the student:

What does it represent?
It represents the number of elements in the container.
Where is it defined?
In class Queue.
How is it implemented?
It returns the value of variable count.
What is its purpose here?
Its value will be compared with the capacity of the container.

What is the purpose of "(size() < CAPACITY)"?
To compare the number of elements in the container with the capacity of the container. If true, the element can be inserted.

What is the purpose of "if (size() < CAPACITY) {... } else ..."?
To insert the element in the Queue if capacity is available.
What if "(size() < CAPACITY)" is eliminated?
An error could be generated when inserting a new element.

In the example, five kinds of questions are illustrated, that is, questions about representation, definition, implementation, purpose, and a what-if question about the effects of eliminating a statement. This constitutes a basic set of questions, which although clearly not exhaustive, covers an important set of possibilities.

Suppose now that from the options offered by the system the student selects the question: "What is the purpose of (size() < CAPACITY)?" After determining an answer to it, the system presents a detailed explanation consisting of a sequence of steps. The presentation of the explanation is controlled by the student, who requests the steps one at a time and can stop the explanation at any point. For illustration purposes we give in fig. 2 the full explanation constructed by the system. In the figure, the interface to the prototype explanation system is shown, with the Code Viewer in the left window, the Container Diagram in the top right window, and the Array Diagram in the bottom right window. Initially, only the code viewer is open.

Let us now examine the explanation in the order in which it is presented to the student. First, a sentential explanation is given about what the condition in question does, which is presented next to the condition, in the code viewer. After reading this part of the explanation, the student may close it by pressing the "OK" button that it contains. If so desired, the student can request the next step in the explanation by pressing the "next" button in the "Explainer" control window at the bottom of the screen. In this case, the next step consists of a diagrammatic explanation corresponding to the sentential explanation just given. To this effect, the container diagram is open showing a container with elements inside, plus entry and exit ports. In addition, the "properties" of the container which are relevant to the condition in question - that is, the number of elements currently contained and the container capacity - are explicitly shown in the diagram, by means of arrows with corresponding names. The third step simply adds a sentential explanation to the container diagram, making explicit what the diagram is showing.

In the fourth step, the purpose of the condition is stated in a sentential explanation in the code viewer, underneath a previous explanation. Finally, if the student still needs additional information, in the fifth and sixth steps the system provides information from an implementation perspective. After opening the array diagram window with a basic depiction of the array, the system presents the relevant aspects of the diagram, that is, variable "count" and constant "CAPACITY." If requested, a final sentential element making explicit what is being shown in the diagram is offered to the student. It should be reiterated that the figure presents a full explanation, and for this reason it appears "busy." But, as previously indicated, the student can stop the explanation at any time and can close any of the individual explanations as well as the diagrams, if desired.

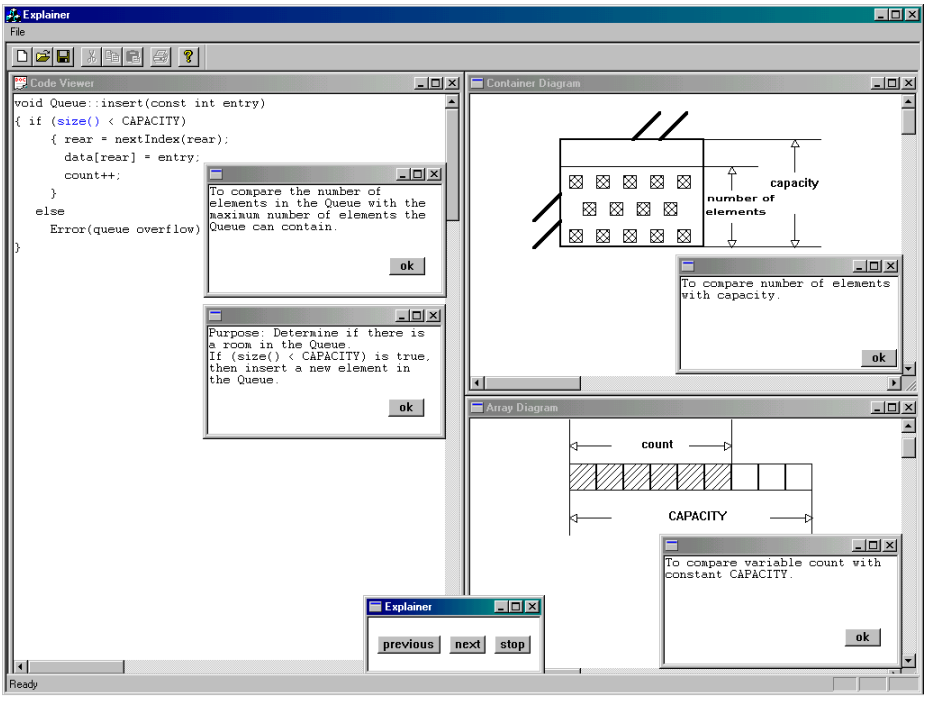


Fig. 2. Explanation of the Purpose of (size() < CAPACITY)

5 Explanation Process

An overall description of the explanation process exemplified above is given below. Three main steps can be identified, that is, to determine candidate questions corresponding to the selected code, to determine an answer to the selected question, and to construct and present an explanation of the answer.

Explain(ADT)
- student selects code element to be explained
- system determines candidate questions taking into account:
  - questions that can be asked of the selected element
  - questions that can be answered with available knowledge
- student selects question among given candidates
- system determines answer to the question
- system computes an explanation of the answer, containing sentential and diagrammatic elements
- an explanation is presented step by step to the student
  - student can examine the explanation, going back and forth, or stopping it at any time

5.1 Knowledge Used and Explanation Methods

The three steps mentioned above make use of a knowledge base represented as a semantic network, which has the following main components. First, there is "container knowledge" that includes knowledge about the concept of a container, both about its structural components and about the subprocesses that operate upon it. For each structural component there are associated templates describing its meaning. For each subprocess there is a decomposition of it in terms of lower level subprocesses, with associated templates describing their purpose. A template here has the conventional meaning of a sentential pattern containing variable elements, plus associated procedures specifying how to compute those elements from the current context.

Second, there is "ADT knowledge," that is, knowledge about the meaning of specific aspects of an ADT implementation. Behind the C++ code there are internal representations of the meaning of particular kinds of declarations, statements, and their recursive composition. These internal representations have templates associated with them, which are utilized during the explanation process. Third, there is "data structure knowledge" containing a representation of lower level data structures, such as arrays, identifying their structural components. Finally, there are "semantic links" connecting ADT knowledge to both container knowledge and data structure knowledge, capturing the "representation" and "implementation" relationships which are crucial to understanding ADTs.

During the explanation process these various kinds of knowledge are used in different ways to carry out the steps of the process. Let us examine in detail how the step that constructs explanations is performed. After a specific question about an element of an ADT is selected, an answer to it in terms of an internal representation is generated. Using the information contained in the answer, an explanation is constructed, under the control of an *explanation method*. Taking into account the kind of question and the kind of code to which it refers, an explanation method is obtained. Such methods consist of a sequence of steps, where each step constructs part of an overall explanation, using sentential and diagrammatic elements. These explanation methods are predefined and have been determined taking into account tutorial and pragmatic considerations, such as "it is important to provide context" and "it is important to explain from several viewpoints," where context and viewpoints are provided by the representation and implementation relationships in which ADTs are involved.

To illustrate the process let us return to the example presented in the previous section. The question posed to the system was about the purpose of the (size() < CAPACITY) condition appearing in the context of the insert function of the Queue ADT. To explain the purpose of a statement or part of a statement in a function there is an explanation method containing the following steps:

Explanation Method(code)
- produce sentential explanation of what code DOES
- generate diagrammatic explanation of what code REPRESENTS
  - produce sentential explanation of the diagrammatic explanation (previous step)
- produce a sentential explanation of the code's PURPOSE
- generate diagrammatic explanation of the IMPLEMENTATION aspects of the code
  - produce sentential explanation of the diagrammatic implementation (previous step)

While, in a strict sense, the purpose of the selected code is explained in the fourth step of the explanation method, the other steps generate contextual information from multiple viewpoints, to make for a more complete explanation. The fourth step produces the following information: "Purpose: Determine if there is room in the Queue. If (size() < CAPACITY) is true, then insert a new element in the Queue." A semantic link, connecting the internal representation of the condition being explained to an appropriate element of the insert subprocess of the container concept, is used to retrieve a template from which the first phrase in the above explanation is constructed. A template associated with the "if-then-else statement" in conjunction with another template obtained via a semantic link connecting the then-part of the statement to a corresponding element of the insert subprocess, is used to generate the second phrase of the above explanation. Fig. 3 illustrates the explanation method and parts of the knowledge involved in handling the above example.

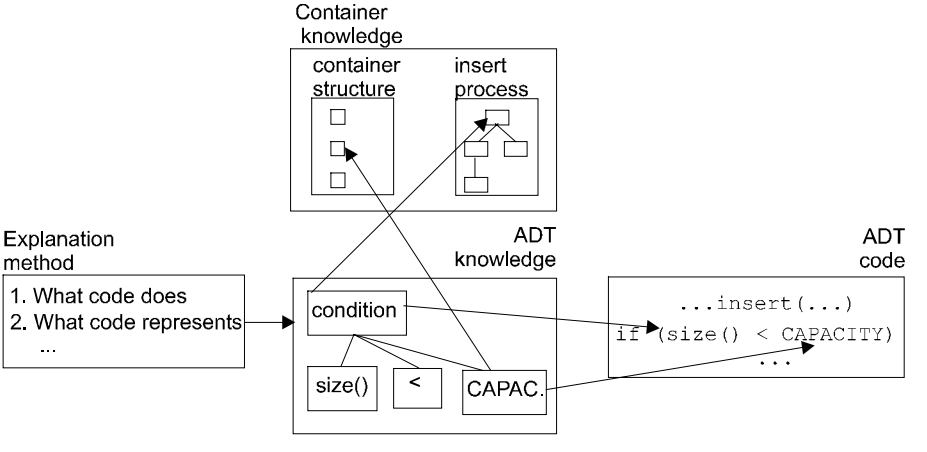


Fig. 3. Explanation Method and Knowledge Used

To summarize, an explanation is computed by the recursive application of templates associated to the internal representation of the code and to the container and data structure parts of the knowledge base, under the guidance of predefined explanation methods.

5.2 A Prototype Implementation

A prototype implementation of the above explanation tool has been developed and continues to be expanded. Both the domain knowledge base and the explanation methods have been implemented in the C++ language following an object-oriented approach. The knowledge base includes a large number of classes describing the container process, the ADTs themselves, and lower level data structures. Currently the system contains knowledge about elementary ADTs and can handle a basic set of questions. The aim is to expand the knowledge base with more advanced ADTs, with additional lower level data structures, and to cover a larger set of questions.

6 Discussion and Conclusions

In this work we have adopted a broad notion of an ADT. By explicitly regarding an ADT as a computational representation of a container process, we can bring to bear the common sense knowledge students have about containers to the understanding of ADTs. Similarly, by explicitly representing the relationships between ADTs and the lower level data structures used in their implementation, we allow the students to apply their knowledge of data structures to the new task. Interleaving sentential and diagrammatic explanations makes it possible to present information from multiple viewpoints in different ways, thus increasing the chances that the student can comprehend the material. For these reasons we can conclude that an explanation tool of this kind has the potential to overcome some of the limitations that characterize books as vehicles for learning about ADTs.

A difficulty with this tool is that its explanations can become repetitive because the explanation methods do not take into account prior student/system interactions. This problem could be alleviated by keeping track of certain aspects of such interactions or, alternatively, by giving the student finer control of the presentation of the explanation. Because the system does not contain a fine grained representation of the logic of ADT functions, a limitation of this tool is that it cannot handle certain kinds of what-if questions which can be very useful to understand how ADTs work.

In the conception of this tool we wanted to strike a balance between flexibility and simplicity. To achieve a good coverage of questions an extensible architecture was required, which we achieved by using an object-oriented approach. To keep the overall system relatively simple, an explanation process was devised that uses as much pre-compiled knowledge as possible, mainly in terms of explanation methods and templates. This deliberate attempt at simplicity is in contrast with most of the work on explanation on ILEs, in particular, with approaches based on planning formalisms (e.g., [12]), which attempt to achieve great flexibility at the cost of substantial complexity.

Once the system has achieved an appropriate coverage of ADTs, we intend to perform experiments with it to determine how useful it is in helping students to achieve a better understanding of ADTs. We should then be able to determine the merits of this approach.

References

1. Andriessen, J., Sandberg, J.: Where is Education Heading and How About AI? Int. J. of Artif. Intell. in Educ. 10 (1999) 130-150.
2. Araya, A.: Suggesting Multiple Design Actions Using Prior Cases. In Procs. of the 7th World Conference on Artificial Intelligence in Education, Washington, D.C. August 16-19, (1995) 373-380.
3. Brusilovsky, P.: Intelligent Learning Environments for Programming: The Case for Integration and Adaptation. In Procs. of the 7th World Conference on Artificial Intelligence in Education, Washington, D.C. August 16-19, (1995) 1-8.
4. Chee, Y.S. and Xu, S.: SIPLeS: Supporting Intermediate SmallTalk Programming through Goal-Based Learning Scenarios. In Procs. of the 8th World Conference on Artificial Intelligence in Education, Kobe, Japan. August (1997) 95-102.
5. Cheng, P. C-H.: Law Encoding Diagrams for Instructional Systems. Int. J. of Artif. Intell. in Educ. 7 (1996) 33-74.
6. Eisenstadt, M., Price, B. A., and Domingue, J.: Redressing ITS Fallacies Via Software Visualization. In Cognitive Models and Intelligent Environments for Learning Programming. Springer-Verlag, Berlin Heidelberg New York (1993).
7. Gonschorek, M. and Herzog, Ch.: Using Hypertext for an Adaptive Help System in an Intelligent Tutoring System. In Procs. of the 7th World Conference on Artificial Intelligence in Education, Washington, D.C. August 16-19, (1995) 274-281.
8. Johnson, W.: Understanding and Debugging Novice Programs. Artificial Intelligence 42 (1990) 51-98.
9. Larkin, J.H. and Simon, H.A.: Why a Diagram Is (Sometimes) Worth Ten Thousand Words. Cognitive Science 11 (1987) 64-100.
10. Lustig, M.: Protocol Driven Hypertext Explanations in a Tutor for Financial Analysis. In Procs. of the 7th World Conference on Artificial Intelligence in Education, Washington, DC. August 16-19, (1995) 290-298.
11. Moore, J.D.: Discourse Generation for Instructional Applications: Making Computer Tutors More Like Humans. In Procs. of the 7th World Conference on Artificial Intelligence in Education, Washington, DC. August 16-19, (1995) 36-43.
12. Salles, P., Bredeweg, B., and Winkels, R.: Deriving Explanations From Qualitative Models. In Procs. of the 8th World Conference on Artificial Intelligence in Education, Kobe, Japan. August (1997) 474-481.
13. Stasko, J., Domingue, J., Brown, M.H., and Price, B.A.: Software Visualization. Programming as a Multimedia Experience. MIT Press, Cambridge London (1998).
14. Tanner, M. and Keuneke, A.: Explanations in Knowledge Systems. The Roles of the Task Structure and Domain Functional Models. IEEE Expert, June (1991) 50-57.
15. Valley, K.: Explanation, Exploration, and Learning: The Use of Expert System Shells in Education. J. of Artif. Intell. in Educ. 3 (1992) 255-273.
16. Van Joolingen, W.R. and De Jong, T.: Design and Implementation of Simulation-Based Discovery Environments: The SMISLE Solution. In Procs. of World Conference on Artificial Intelligence in Education, Washington DC, August 16-19, (1995) 99-106.
17. Weber, G.: Providing Examples and Individual Remindings in an Intelligent Programming Environment. In Procs. of World Conference on Artificial Intelligence in Education, Washington DC, August 16-19, (1995) 477-484.

Dialogue Structure Influence Over Anaphora Resolution*

Patricio Martínez-Barco and Manuel Palomar

Departmento de Lenguajes y Sistemas Informáticos
Universidad de Alicante
Carretera de San Vicente del Raspeig - Alicante - Spain
Tel. +34965903653 Fax. +34965909326
{patricio, mpalomar}@dlsi.ua.es

Abstract. Nowadays, anaphora resolution systems obtain satisfactory results when applied to restricted domains or specific texts. The problem arises when changes of the text (e.g. from one domain to another), or changes of the kind of text (e.g. from non-dialogue texts to dialogues) are attempted. In this case, results are not as good as people hope. In this paper, this problem and the relationship between anaphora resolution and dialogue structure is presented. Thus, we start with an initial anaphora resolution system applied to non-dialogue texts and we study its impact in dialogue texts. From this system, we have developed some experiments to demonstrate the need for using dialogue structure information in order to solve the anaphora in this kind of corpora.

1 Introduction

Anaphora understanding is an important process in any NLP system, and it is among the toughest problems to solve in Computational Linguistics and NLP. Anaphora could be classified in many different ways, depending on the particular criteria one chooses to employ. Regarding, for example, the element that carries out the reference (or anaphor), clear distinctions should be made between pronominal anaphora, adjectival anaphora, definite descriptions, one-anaphora, surface-count anaphora, verbal-phrase anaphora and time and/or location references.

Several systems have been developed in order to solve the anaphora in different domains, using several techniques. However, the anaphora resolution problem changes according to the domain and to the language. Occurrence and variety of anaphora is different from dialogues to discourses, or from Spanish to English. This matter means that systems working in specific domains have a lot of problems when they are transferred to other domains. In this paper, changes that must be performed in order to adapt an anaphora resolution system for pronominal and adjectival anaphora in non-dialogue Spanish texts into

* This paper has been supported by the Spanish Government under grant HB1998-0068.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 515–525, 2000.

an anaphora resolution system for Spanish dialogues will be shown. Moreover, experiments will be performed that will demonstrate the need for using dialogue structure information in order to solve and understand the anaphora in this kind of corpora.

These experiments have been evaluated on a dialogue corpus provided by the project *Basurde*[1]. This corpus has 200 dialogues containing conversations between a telephone operator of a railway company and users of this company. 40 of them were randomly selected and POS-tagged for the evaluation. Several experiments have been performed on this evaluation in order to define the adequate combination of different kinds of knowledge.

In the following section, the use of constraints and preferences as an approach to anaphora resolution will be presented. This is followed by a demonstration of the need for using information about dialogue structure in order to solve the anaphora. Finally, some conclusions about our work in progress will be shown.

2 Constraints and Preferences as an Approach to Anaphora Resolution

According to Dahlbäck, efforts made so far towards anaphora resolution can be divided into two basic approaches [Dah91]: Traditional and Discourse-oriented. The traditional approach generally depends on linguistic knowledge. In the discourse-oriented approach, however, the researcher tries to model the complex discourse structure. Anaphora, accepted as discourse phenomena, is resolved with the help of that complex structure.

Among the traditional approaches the following works stand out: Mitkov [Mit98], Baldwin [Bal97] and Ferrández et al. [FPM99], all of which are based on a combination of linguistic knowledge (lexical, morphological, syntactic and/or semantic) for the resolution of anaphora. These approaches apply linguistic knowledge in the way of constraints and preferences following the works of Carbonell & Brown [CB88] and Rich & LuperFoy [RL88]. In these works, constraint and preference systems are proposed as a technique for combining several information sources.

These approaches are based, intuitively, on the following three steps: a) anaphoric accessibility space definition, b) application of constraint system, and c) application of preference system.

A constraint and preference system must define, firstly, the anaphoric accessibility space. That is, it must obtain a list with all the possible candidates that can be the anaphor antecedent. Then, the system will define the text spaces where the antecedent can be found. This step has a great importance in the remaining process because definitions of anaphoric accessibility space which are too short cause the removal of valid antecedents for the anaphor. On the other hand, definitions of anaphoric accessibility space which are too large cause large

[1] BASURDE: Spontaneous-Speech Dialogue System in Limited Domains. CICYT (TIC98-423-C06).

candidate lists, where failure probabilities in anaphora resolution are increased. Usually, anaphora resolution systems based on linguistic knowledge define an accessibility space using n previous sentences to the anaphor, where n is variable according to the kind of the anaphora.

Once the list of possible candidates is defined, several constraints are applied in order to remove incompatible antecedents. The constraint system will consist of conditions with 100% fulfillment probability. So, any candidate not fulfilling such conditions will be considered an imposible antecedent for the anaphor. Lexical, morphological, syntactical and semantic information is traditionally used in order to define due constraints.

Finally, after removing incompatible candidates, and when the list has more than one antecent, preferences are applied in order to choose only one antecedent. In this case, unlike constraints, preferences have a fulfillment probability of less than 100%. However, it is well-known that candidates fulfilling a preference have more probability of being the antecedent than others not fulfilling it. The preference system must be designed bearing in mind that only one candidate must remain at the end. This final candidate will be proposed as the antecedent for the anaphor. Lexical, mophological, syntactic and semantic information is usually used in order to define the preference system.

Works like Mitkov [Mit98] and Ferrández et al. [FPM99] show anaphora resolution systems based on constraints and preferences to have successful results when they are applied to non-dialogue texts. However, these works do not show an adequate proposal for the anaphoric accessibility space. Furthermore, these approaches lack consistency in the treatment of other kinds of texts like, for example, dialogues.

In the next section, the importance of defining an adequate accessibility space in order to solve the anaphora will be demonstrated. For this, we will start applying the constraint and preference system introduced by Ferrández et al. [FPM99] to dialogue treatment. This constraint and preference system has been demonstrated to be adequate for pronominal and adjectival anaphora[2] in discourse. From this system, information about dialogue structure will be applied allowing us to demonstrate the influence of this structure, using it as an anaphoric accessibility space as well as a preference in order to solve anaphora in Spanish dialogues.

3 Dialogue Structure Information as a Preference

In order to show the importance of dialogue structure in anaphora resolution, a constraint and preference system has been defined and it has been integrated as an independent module in a Natural Language Processing (NLP) system. This NLP system consists of a morphologic analyzer, a POS tagger, a partial parser and several modules for linguistic phenomena resolution. One of these modules is the anaphora resolution module. This module enjoys the advantage

[2] Spanish adjectival anaphora agrees with English one-anaphora, but in Spanish, the word *one* is omitted.

of being flexible and available to any spoken dialogue system. Over this NLP system, the precision of our anaphora resolution system will be evaluated using 40 spoken dialogues that have been obtained by means of the transcription of conversations between a telephone operator of a railway company and users of the company. In these dialogues, users ask for information about the company service. A POS-tagged process was carried out over these 40 dialogues, and then, a manual dialogue structure annotation was performed. Five of them were randomly selected for the training of the manual annotation process, and the remaining 35 were reserved in order to carry out the final evaluation.

On this evaluation, several experiments have been carried out on this with changes in the constraint and preference system in order to define the configuration[3] that causes optimum precision. These experiments are shown in summary table 1.

Table 1. Experiment summary

	Used preferences																						Precision	
Experiment	Pronominal														Adjectival								Pron.	Adj.
number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8	9	10	%	%
0	X	X			X	X	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	59.0	23.7
1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	54.1	68.4
2	X	X	X	X											X	X	X	X							62.3	65.8
3	X	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X			X	67.2	76.3
4	X	X	X	X		X	X	X	X	X	X			X	X	X	X	X		X	X			X	72.1	68.4
5	X	X	X	X			X	X	X	X	X			X	X	X	X	X	X		X			X	73.8	76.3
6	X	X	X	X				X	X	X	X			X	X	X	X	X	X	X				X	73.8	78.9
7	X	X	X	X					X	X	X			X	X	X	X	X	X	X				X	73.8	78.9
8	X	X	X	X						X	X			X	X	X	X	X	X	X				X	72.1	76.3
9	X	X	X	X					X		X			X	X	X	X	X	X	X				X	68.9	78.9
10	X	X	X	X					X	X				X	X	X	X	X	X	X				X	73.8	78.9

3.1 Experiment 0 (Basic Experiment): Linguistic Information Only

We started from the initial system. This system is based on linguistic information only, and its results have been successfully tested over a non-dialogue corpus obtaining a precision of 82% for pronominal anaphora resolution (we have no information about precision in the case of adjectival anaphora). Next, this constraint and preference system was applied over the dialogue corpus.

This basic configuration contained the following constraint and preference system, and the following anaphoric accessibility space.

[3] The term *configuration* is used in order to define the constraint and preference set that makes up the system in a concrete instant of the experiment process.

Anaphoric Accessibility Space. In pronominal anaphora resolution, an anaphoric accessibility space using the three previous sentences/utterances to the anaphor was defined. In adjectival anaphora, the space was increased to four sentences/utterances.

Constraints

- In the case of pronominal anaphora:
  1. Morphological agreement: gender, number and person
  2. C-command constraints
- In the case of adjectival anaphora:
  1. Morphological agreement: gender
  2. C-command constraints

Preferences

- In the case of pronominal anaphora:
  1. Candidates that are in the same sentence/utterance as the anaphor
  2. Candidates that are in the previous sentence/utterance
  3. EMPTY (at the moment)
  4. EMPTY (at the moment)
  5. Candidates that are proper nouns or indefinite NPs
  6. If the anaphor is a personal pronoun, then preference for proper nouns
  7. Candidates that have been repeated more than once
  8. Candidates that have appeared with the verb of the anaphor more than once
  9. Candidates that are in the same position as the anaphor with reference to the verb (before or after)
  10. Candidates that are in the same syntactic constituent (they have the same number of parsed constituent as the anaphor)
  11. Candidates that are not in CC
  12. Candidates most repeated in the text
  13. Candidates most appeared with the verb of the anaphor
  14. The closest candidate to the anaphor
- In the case of adjectival anaphora:
  1. Candidates that are in the same sentence/utterance as the anaphor
  2. Candidates that are in the previous sentence/utterance
  3. EMPTY (at the moment)
  4. EMPTY (at the moment)
  5. Candidates that share the same kind of modifier (e.g. a prepositional phrase)
  6. Candidates that share the same modifier (e.g. the same adjective 'red')
  7. Candidates that agree in number
  8. Candidates more repeated in the text
  9. Candidates appearing more with the verb of the anaphor
  10. The closest candidate to the anaphor

Discussion. According to this first configuration, a corpus evaluation was performed obtaining a precision of 59.0% for pronominal anaphora resolution and a precision of 23.7% for adjectival anaphora. As can be appreciated, the obtained result is very low for pronominal anaphora, and it is extremely poor in adjectival anaphora. Once the failures were evaluated, the following conclusions were arrived at: the defined anaphoric space was too short for the considered anaphora, and this space was defined in an arbitrary way, without regard for the relationship between anaphora and dialogue structure. Consequently, we proposed the changes that can be seen in experiment 1.

3.2 Experiment 1: Linguistic Information Plus Dialogue Structure Information (First Approach)

In this experiment, the definition of anaphoric accessibility space was changed, using in this case, the information that dialogue structure provides according to the proposal of Martínez-Barco et al. [MBPFM99], as well as the preferences affected by this definition.

Anaphoric Accessibility Space. The adjacency pair[4] and the topic of the dialogue were used in order to define the anaphoric accessibility space. Concretely, we defined an anaphoric accessibility space by means of the adjacency pair of the anaphor, the previous adjacency pair of the anaphor, adjacency pairs containing the adjacency pair of the anaphor, and finally, the main topic of the dialogue (for pronominal as well as adjectival anaphora).

Preferences. We considered the same set of preferences as in experiment 0, but in this case, preferences 1 to 4 were replaced by the following new preferences regarding the new anaphoric accessibility space (for pronominal and adjectival anaphora):

1. Candidates that are in the same adjacency pair as the anaphor
2. Candidates that are in the previous adjacency pair to the anaphor
3. Candidates that are in some adjacency pair containing the adjacency pair of the anaphor
4. Candidates that are in the topic

Remaining preferences stayed as in experiment 0.

[4] As in Fox's research [Fox87] about the relationship between anaphora and discourse structure, we use conversational analysis [SSJ74] to define the hierarchical structure of dialogue. In accordance with this approach, the adjacency pair is the basic unit used to identify the structural organization of the discourse (initiative and reaction).

Discussion. After this experiment, precision rates of 54.1% for pronominal anaphora and 68.4% for adjectival anaphora were obtained. A considerable increase is noticed in the resolution of adjectival anaphora with only changing the accessibility space. That is due to the fact that adjectival anaphora needed a larger space than the previous one. Moreover, a slight decrease in pronominal anaphora resolution can be noticed. After studying it, we have seen how the number of candidates grows as the accessibility space grows. The problem appears when wrong preferences are used. If the number of candidates is high and the preferences are not correctly defined, the risk of taking wrong preferences is greater. So, the probability of success decreases. Thus, our experiments were concentrated on redefining the preference set removing those preferences that, being incorrectly defined, caused erroneous results when applied to dialogues.

3.3 Experiment 2: Dialogue Structure Information Only

Preferences. In this experiment, pronominal anaphora preferences 5 to 14, and adjectival anaphora preferences 5 to 10 were removed. This change was made in order to test the system's performance when linguistic information is removed and only dialogue structure information is born in mind (preferences 1 to 4).

Discussion. After removing all the linguistic preferences, precision rates of 62.3% for pronominal anaphora, and 65.8% for adjectival anaphora resolution have been achieved. These results still being low demonstrate that dialogue structure information does not produce satisfactory results when applied alone. Thus, the next experiment was performed starting from experiment 1, and several variations in the preference system were carried out separately.

3.4 Experiment 3: Linguistic Information Plus Dialogue Structure Information (Second Approach)

Preferences. In this experiment, all the preferences from experiment 1 were used except 12 and 13 in pronominal anaphora, and 8 and 9 in adjectival anaphora.

Discussion. Usually, information about repeated candidates is inserted into the preference system in order to achieve knowledge about the main entities of the dialogue. However, in the previous experiment, information about the main topic of the dialogue has been included, and so, preferences about repeated candidates are not needed. Thus, pronominal anaphora preferences 12 and 13, and adjectival anaphora preferences 8 and 9 (candidates most repeated in the text and candidates that most appeared with the verb of the anaphor) were removed. After this experiment the results were improved, obtaining a precision of 67.2% for pronominal anaphora resolution and 76.3% for adjectival anaphora resolution. Consequently, these preferences were not used in the following experiments.

3.5 Experiment 4: Linguistic Information Plus Dialogue Structure Information (Third Approach)

Preferences. In this case, all the preferences from experiment 3 were used except preference number 5 in pronominal anaphora and preference number 5 in adjectival anaphora.

Discussion. In this experiment, pronominal anaphora preference number 5 (candidates that are proper nouns or indefinite NPs) and pronominal anaphora preference number 5 (candidates that share the same kind of modifier) were removed in parallel. After that, we noticed the following result: pronominal anaphora resolution had improved achieving a precision of 72.1%, whereas adjectival anaphora resolution decreased to 68.4%. In the case of adjectival anaphora, preference for antecedents using the same structure of modifier was demonstrated to be meaningful. Thus, the preference was included in the remaining experiments. Nevertheless, pronominal anaphora preference for proper nouns or indefinite noun phrases made the results worse, consequently, this preference was not used in other experiments.

3.6 Experiment 5: Linguistic Information Plus Dialogue Structure Information (Fourth Approach)

Preferences. We started from the preference set defined in experiment 4, removing preference number 6 for adjectival and pronominal anaphora resolution, and integrating preference number 5 for adjectival anaphora resolution again.

Discussion. In this experiment, preference number 6 for pronominal anaphora (preference for proper nouns, if personal pronoun) and preference number 6 for adjectival anaphora (candidates that share the same modifier) were removed. In this case, we noticed that results for pronominal anaphora were improved, achieving a precision of 73.8%, whereas adjectival anaphora results stayed at 76.3%. After studying the case, we deduced that preference for proper nouns causes errors in the domain where the experiment was performed due to the existence of place names where this preference is not valid. In this case, we decided to remove preference number 6 for pronominal anaphora in the remaining experiments. However, our observation over other corpora leads us to include preference number 6 for adjectival anaphora, because we consider this information can be an important factor in the future.

3.7 Experiment 6: Linguistic Information Plus Dialogue Structure Information (Fifth Approach)

Preferences. We started from the preference set defined in experiment 5, removing, in this case, preference number 7 for pronominal and adjectival anaphora resolution, and integrating preference number 6 for adjectival anaphora again.

Discussion. A test in parallel was performed again removing preference number 7 in both kinds of anaphora. Preference number 7 for pronominal anaphora (candidates that have been repeated more than once) is used in systems lacking information about the main topic of the dialogue in order to measure the level of the candidate's salience on the text. However, when information about the topic is included in the system, this preference becomes meaningless. After removing it, a precision of 73.8% for pronominal anaphora resolution was obtained. On the other hand, preference number 7 for adjectival anaphora (candidates that agree in number) that provides good results in non-dialogue texts, lacks justification for using it in dialogues. In this case, when this preference was removed, a precision of 78.9% was achieved. Because results in pronominal anaphora were not worse, and results in adjectival anaphora were clearly better, we decided to remove both preferences in further experiments.

At this stage, the experiment process for adjectival anaphora resolution has been concluded because all the preferences have been evaluated, and the optimum configuration has been achieved. Nevertheless, the experiment process carried on evaluating preferences for pronominal anaphora.

3.8 Experiment 7: Linguistic Information Plus Dialogue Structure Information (Sixth Approach)

Preferences. We started from the preference set defined in experiment 6, removing preference number 8 for pronominal anaphora resolution. The preference set for adjectival anaphora was the same as in experiment 6.

Discussion. After removing preference number 8 for pronominal anaphora (candidates that have appeared with the verb to the anaphor more than once) the same preference of 73.8% was obtained. Thus, this preference was not used for further experiments. Besides, its usefulness has not been justified properly. In this case, a precision of 78.9% for adjectival anaphora was obtained.

3.9 Experiment 8: Linguistic Information Plus Dialogue Structure Information (Seventh Approach)

Preferences. We started from the preference set defined in experiment 7, removing preference number 9 for pronominal anaphora resolution.

Discussion. After removing preference number 9 for pronominal anaphora (candidates that are in the same position as the anaphor with reference to the verb -before or after-) a precision of 72.1% was obtained, being lower than the best value obtained previously. Besides, the precision for adjectival anaphora decreased to 76.3% due to carried errors. Thus, the preference was integrated into the system again.

3.10 Experiment 9: Linguistic Information Plus Dialogue Structure Information (Eighth Approach)

Preferences. We started from the preference set defined in experiment number 8, where preference number 10 was removed, and preference number 9 was integrated again, both in the case of pronominal anaphora.

Discussion. Removing preference number 10 for pronominal anaphora (candidates that are in the same syntactic constituent) caused a decrease in precision to 68.9%, while precision for adjectival anaphora stayed at 78.9%. Thus, preference 10 was considered suitable for pronominal anaphora, and it was integrated again.

3.11 Experiment 10 (Final): Linguistic Information Plus Dialogue Structure Information (Ninth Approach)

Preferences. We started from the preference set defined in experiment number 9, where preference number 11 was removed, and preference number 10 was integrated again, both in the case of pronominal anaphora.

Discussion. After removing preference 11 (candidates that are not in CC), the system solved 73.8% of pronominal anaphora, being the best value achieved. Besides, the precision of 78.9% for adjectival anaphora stayed the same. We decided to remove preference number 11, because its usefulness had not been justified properly. Then, after using up all the possibilities, and due to this being the minimum set of preferences, we considered this to be the optimum configuration.

4 Conclusion

In this paper we have demonstrated that:

- the definition of an anaphoric accessibility space based on dialogue structure, and the set of preference according to this structure, helps anaphora resolution.
- traditional anaphora resolution systems are not easily transferable to other kinds of texts, as the experiments developed have demonstrated.
- anaphora resolution in dialogues requires an hybrid system able to combine linguistic information plus main topic information. In this case, the task that requires a greater effort is to find a method that combines both approaches.

For this, our work in progress focuses on developing these three ideas. More information about this work can be obtained in [MBMA$^+$99] and [MBPFM99].

References

Bal97. B. Baldwin. CogNIAC: high precision coreference with limited knowledge and linguistic resources. In *Proceedings of ACL/EACL workshop on Operational factors in practical, robust anaphora resolution*, July 1997.

CB88. J.G. Carbonell and R.D. Brown. Anaphora resolution: a multi-strategy approach. In *Proceedings of 12th International Conference on Computational Linguistics, COLING'88*, pages 96–101, 1988.

Dah91. N. Dahlbäck. *Representations of Discourse-Cognitive and Computational Aspects*. PhD thesis, Department of Computer and Information Science, Linköping University, Linköping, Sweden, 1991.

Fox87. B. Fox. *Discourse Structure and Anaphora.* Written and conversational English. Cambridge Studies in Linguistics. Cambridge University Press, Cambridge, 1987.

FPM99. A. Ferrández, M. Palomar, and L. Moreno. Importance of different kinds of knowledge for pronominal anaphora resolution. *Computational Linguistics (Submitted)*, 1999.

MBMA$^+$99. P. Martínez-Barco, R. Muñoz, S. Azzam, M. Palomar, and A. Ferrández. Evaluation of pronoun resolution algorithm for Spanish dialogues. In *Proceedings of the Venezia per il Trattamento Automatico delle Lingue, VEXTAL*, November 1999.

MBPFM99. P. Martínez-Barco, M. Palomar, A. Ferrández, and L. Moreno. Anaphora resolution algorithm in spoken dialogue systems. *Natural Language Engineering. Special Issue on Best Practice in Spoken Language Dialogue Systems Engineering (Submitted)*, 1999.

Mit98. R. Mitkov. Robust pronoun resolution with limited knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, COLING-ACL'98*, August 1998.

RL88. E. Rich and S. LuperFoy. Anaphora architecture for anaphora resolution. In *Proceedings of Second Conference on Applied Natural Language Processing, ANLP*, 1988.

SSJ74. H. Sacks, E. Schegloff, and G. Jefferson. A simplest systematics for the organization of turn taking for conversation. *Language*, 50(4):696–735, 1974.

Processing of Spanish Definite Descriptions*

Rafael Muñoz, Manuel Palomar, and Antonio Ferrández

Grupo de investigación en Procesamiento del Lenguaje y Sistemas de Información
Departamente de Lenguajes y Sistemas de Informáticos. Universidad de Alicante
Apartado 99. 03080 Alicante, Spain
{rafael,mpalomar,antonio}@dlsi.ua.es

Abstract. Natural language processing systems require a suitable linguistic co-reference resolution. Many works have been focused on pronominal or adjective anaphora resolution in English and Spanish texts. However, few of them have been focused on the resolution of definite descriptions. Traditionally, anaphora resolution was not usually treated properly as a linguistic phenomenon, due fundamentally to the lack of linguistic knowledge or suitable sources of information for its resolution. In this work, a method for the resolution of the *identity co-reference* produced by definite descriptions in Spanish texts is presented. This method is based on the linguistic knowledge acquired by POS-tagger, synonymous dictionary and relationships between names and verbs resources. This method uses a system of restrictions and preferences in order to obtain the correct antecedent. The method achieves a precision of 95% for the identification of non-anaphoric and anaphoric uses of definite descriptions. Moreover, the method achieves a precision of 80% for the reference resolution task. In addition, a profound study of linguistic characteristics of definite descriptions in a Spanish corpus has been made. From this study, a classification of the different definite description types has been obtained.

1 Introduction

Anaphora resolution is one of the most important and difficult problems to be solved in Computational Linguistics. It is a fundamental component in any natural language processing system. In order to solve this linguistic phenomenon, this anaphora resolution component can provide information that does not explicitly appear in texts. For example, an information extraction system with no anaphora resolution component cannot extract an entity when a relevant entity is replaced by an anaphoric expression. Furthermore, in a machine translation system it is important to solve the antecedent of a pronoun when the source and target language do not have genre concordance, e.g. English pronoun *they* and *its* equivalent Spanish pronouns *ellos/as*. Finally, it is easy to find examples in a dialogue system in which speakers replace entities previously mentioned by pronouns and definite descriptions.

* This paper has been supported by the Comisión Interministerial de Ciencia y Tecnología, project number TIC97-0671-C02-02.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 526–537, 2000.

There are several grammatical categories with referential properties such as pronouns, adverbs, adverbial adjuncts and definite descriptions[1]. This paper focuses on resolving definite descriptions.

This paper is organized as follows. In section two, a study of several algorithms that solve definite descriptions is made. In section three, a classification of definite descriptions that are used in written texts is presented. This classification is based on their referential or non-referential characteristics. Referential definite descriptions are also classified from the kind of knowledge that is used to establish the relationship between anaphoric expressions and antecedents. In section four, an algorithm to solve definite descriptions in unrestricted Spanish text is proposed. This algorithm deals with identity co-references[2] between antecedent and anaphoric expression (definite description). In section five, a system's evaluation from a fragment of LEXESP[3] corpus is made.

2 Background

Processing of definite descriptions has been pointed out by several authors. Among them, two methods can be emphasised. Firstly, a pronominal anaphora and definite description processing algorithm is described by Kameyama in [5]. This algorithm is applied to the information extraction system FASTUS [4]. This information extraction system works in a general geo-political business domain, in which its anaphora resolution algorithm only solves *identity co-reference* without evaluating other co-reference types as *parts of, set - subset and set - member.* The algorithm is based on three factors:

a) Accessible text region.
b) Semantic consistency.
c) Dynamic syntactic preference.

This algorithm uses a searching window *(accessible text region)* for each anaphoric expression type. The window size was arbitrarily set to ten sentences for definite descriptions and three sentences for pronouns. *Semantic consistencies* between anaphoric expression and its potential antecedent are based on *number consistency* (anaphoric expression and antecedent must be consistent in number, singular or plural), *sort consistency* (anaphoric expression sort must be either equal or subsume antecedent sort) and *modifier consistency. Dynamic syntactic preferences* are the last factor that this algorithm uses. There are three dynamic syntactic preferences: (1) The preceding part of the same sentence in the

[1] Definite description is defined as a noun phrase headed by the definite article, such as *the dog*

[2] Identity co-reference occurs when an anaphoric expression can be replaced by antecedents without changing the sense of the sentence

[3] LEXESP is a Spanish corpus. This corpus has about 5 million of tagged words developed by Psychology Department from University of Oviedo, Computational Linguistic Group of University of Barcelona and Language Treatment Group from Technical University of Cataluña.

left-right order. (2) The immediately preceding sentence in left-right order. (3) Other preceding sentences within the searching window in the right-left order. Also, alias and acronym references are solved by the algorithm, although results are not shown. This algorithm achieved 46% recall[4] for definite descriptions, 62% recall for pronouns and 69% recall for proper nouns. In general, the performance of this system achieved a recall of 59% and precision[5] of 72% in the evaluation of 30 newspaper articles.

Secondly, a system that resolves definite description is presented by Vieira and Poesio in [11] and [10]. This system solves references between definite descriptions and antecedents with either the same head noun or a semantic relationship (synonym, hyperonym, hyponym). Clark, in [2], named *bridging references* the uses of definite descriptions whose antecedents have a different head noun. Vieira and Poesio algorithm executes four tests for identifying new discourse descriptions before trying to find out an antecedent. If these tests fail, the system will look for an antecedent with same head as the anaphoric expression. Finally, the system applies several heuristic rules in order to look for semantic relations (*synonym, hyponym* and *meronym*) between head nouns. This algorithm achieved 62% recall and 83% precision in solving direct anaphora (same head noun). Bridging descriptions were evaluated by hand, and 61 relations of 204 in the corpus were achieved. These works use 20 parsed texts selected from the *Penn Tree Bank* corpus.

3 Classification of Definite Descriptions

Several taxonomies as Cristopherson [1], Hawkins [3], Prince [9] and Poesio & Vieira [8] have been carried out for definite descriptions in the English language. These taxonomies are usually based on relations between the anaphoric expression (definite descriptions) and the antecedent. Moreover, Poesio and Vieira's classification distinguishes an additional category using information about knowledge that speaker and listener have about the object that they are talking about. In the Spanish language, no definite description taxonomy has been accomplished. One of the main differences between English and Spanish definite descriptions is the usual omission of the determiner *the* in the English language, whereas in Spanish it is usual to add it to all noun phrases (determiner *el/la*). For example, the Spanish sentence *La generosidad es uno de los valores más importantes* is translated into English as $\emptyset_{the}$ *Charity is one of the most important values*, where $\emptyset_{the}$ symbol points the position of the determiner the. This fact is due to the fact that definite descriptions in English are only used to refer to a previously mentioned entity. Therefore, the number of Spanish definite descriptions is greater than English ones.

In the taxonomy presented by Poesio and Vieira in [8], all definite descriptions can be classified, although they cannot be classified from the kind of know-

[4] Recall is the quotient between correctly solved and number of occurrences.

[5] Precision is the quotient between number of correctly solved and number of processed occurrences.

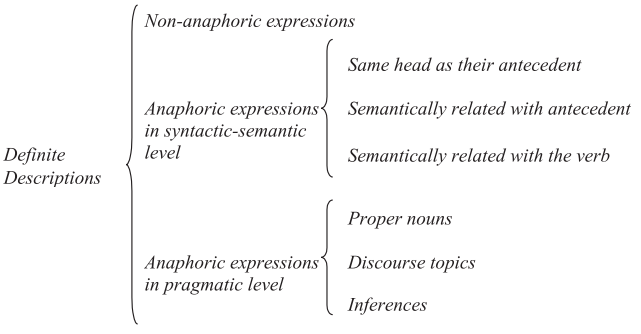


Fig. 1. *Definite description's classification*

ledge that is needed to solve the reference between anaphoric expression and its antecedent.

In Figure 5 a definite description (*DD*) classification obtained from the LEXESP corpus is shown. This study has been accomplished according to the linguistic level of the kind of knowledge needed to solve the anaphoric expression. There are four linguistic levels: phonologic, morphologic, syntactic-semantic-textual and pragmatic level.

Next, each category in Figure 5 is going to be described in full detail:

Non-anaphoric expressions. Within this category, those definite noun phrases that have no reference meaning with other previously mentioned entities are included. These noun phrases usually introduce a new entity that can be referenced by other noun phrases. For example, let us suppose a text that starts with the sentence *La casa de la playa es bonita (The house on the beach is beautiful)*. In this case, the definite description *The house on the beach* has not previously appeared in discourse, therefore, it introduces a new entity.

Anaphoric expressions in syntactic-semantic level. In this category, those definite noun phrases that reference other previously mentioned entities are included. According to the kind of syntactic-semantic relation that is established between anaphoric expression and its antecedent, we can distinguish between the following types:

Anaphoric expressions with the same head as their antecedent. That is to say, those noun phrases that share their head noun. For example, in the following text: *La casa de la playa es bonita ... La casa era de un médico (The house on the beach is beautiful ... It was the house of a doctor)*, the underlined noun phrases share the same head noun, i.e. *casa (house)*. Those Spanish noun phrases with their head omitted[6] are also included in this category. For example, *La del médico tiene piscina (The doctor's one has a swimming-pool)*.

[6] This kind of noun phrases are quite usual in Spanish texts. Its equivalent in English would be one-anaphora.

Anaphoric expressions semantically related with their antecedent. It occurs when the head noun of the definite description is a *synonym*, *hyponym* or *meronym* of its antecedent's head. For example, in *Los soldados irrumpieron en el poblado. Estos hombres valientes... (The soldiers burst into the village. These brave men...)*, the definite description *estos hombres valientes (these brave men)* refers to *los soldados (the soldiers)* since *soldiers* is an hyponym of *men*.

Anaphoric expressions semantically related with the verb. This kind of expressions refers to an antecedent that appears in a sentence whose verb means the action developed by the description. For example, in *Juan vende su casa a Fernando. El vendedor pagará las costas (Juan sells his house to Fernando. The seller will pay the expenses)*, the definite description *el vendedor (the seller)* refers to *Juan* since it is the agent of the action of the verb *to sell*.

Anaphoric expressions in pragmatic level. This category includes definite descriptions that indirectly refer to their antecedents. In order to solve these references the knowledge of the world is needed. We can distinguish between the following kinds of references:

Proper nouns, i.e. definite descriptions whose antecedent is a proper noun. Moreover, they usually mean some characteristics of their antecedent. For example: *Bill Clinton and The President of US*.

Discourse topics, i.e. definite descriptions that refer to the discourse topic of the text in which anaphora occurs. For example, in a sports text, we can usually make references to sports-teams.

Definite descriptions that can be inferred, i.e. when the anaphoric expression has a complex relationship with its antecedent (cause-effect relation, cause-consequence relation, etc.). For example: *Los ataques eran bastante usuales. Las víctimas estaban por todas partes. (The attacks were quite usual. The victims were everywhere)*, where the victims are a consequence of the attacks.

Syntactic-semantic-textual levels include definite descriptions named by Poesio and Vieira as the *same head anaphora* and some of their *associative uses* (*synonym, hyponym, meronym* and *events*) in [8]. The world knowledge is not needed in order to solve *anaphoric expressions in syntactic-semantic level*. It can

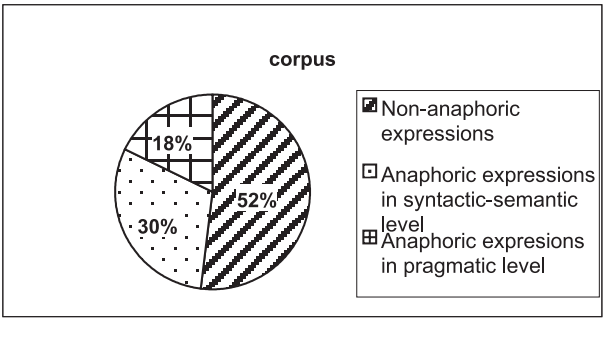


Fig. 2. *Distribution of definite descriptions in LEXESP corpus*

be solved by means of syntactic knowledge and semantic knowledge (e.g. knowledge obtained from WordNet or EuroWordNet that provide us with information about *synonym, hyponym* or *meronym*).

In Figure 2, the distributions of definite descriptions in a fragment of LEXESP corpus are shown. It should be remarked that 52% of definite descriptions do not refer to any previous noun phrase, i.e. they introduce a new entity in discourse. The 18% of definite descriptions need world knowledge to find out their antecedent (pragmatic level) and the remaining 30% use syntactic and semantic knowledge (synonym, relation between verb and noun, etc.).

Among anaphoric expressions in syntactic-semantic-textual level, such Figure 3 shows, the 81% of definite descriptions have the same head-noun as their antecedents. The 17% has a synonymy-hyperonymy relation between both heads, and for the remaining 2% there is a relation between antecedent's verb and definite description.

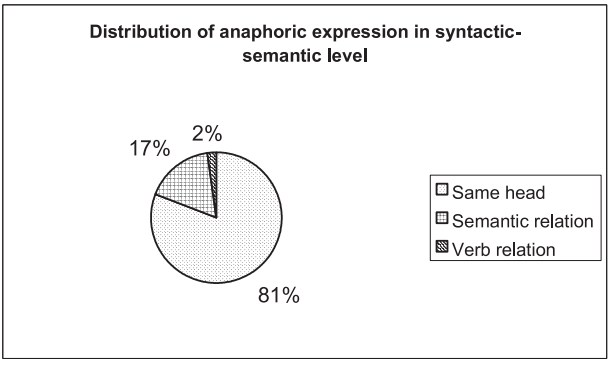


Fig. 3. *Distribution of anaphoric expressions in syntactic-semantic-textual level*

4 Our Method

As we can see in the previous taxonomy, world knowledge is necessary to solve anaphoric expressions in pragmatic level. Since this kind of knowledge is computationally expensive, in this paper, definite descriptions that introduce a new entity and anaphoric expressions in syntactic-semantic-textual level are the only ones that are going to be solved. Those in the pragmatic level are going to be considered as expressions that introduce a new entity in discourse.

In Figure 4, an algorithm to solve definite descriptions is shown. This algorithm takes as input the slot structure (*SS*) of the sentence and a list of antecedents (*LA*) that have appeared in previous sentences. It returns a list of antecedents (*LASal*) as output in which the definite description is added if it introduces a new entity, or otherwise its antecedent is added. The steps of the algorithm are the following:

- The detection of the definite description in the text.
- The application of restrictions on the list of antecedents.
- The application of preferences on the list of remaining antecedents after restrictions.
- The search of the following definite description.

Next, each step of the algorithm is going to be described in full detail.

```
Solving DD
        use SS, LA
        return LASal
 while not end of SS
    Look for DD into SS
    If NP is a DD then
      Applying restrictions to Antecedents List
      Appllying preferences to Antecedenst List after restrictions
      If not empty Antecedents Lists after preferences then
        Adding Antecedent to Antecedents List LA, LASal)
      else
        Adding DD to Antecedents Lists
     else
       Adding NP to Antecedents Lists
End while
```

Fig. 4. *Algorithm to solve definite descriptions*

4.1 The Detection of the Definite Description in the Text

The algorithm for the resolution of definite descriptions starts searching these definite descriptions in sentences. This algorithm considers definite descriptions as noun phrases that begin with a definite or demonstrative article. Otherwise, if the algorithm detects others noun phrase types are introduced in the list of antecedents they are considered as new entities.

The algorithm takes the output of the partial parser *SUPP* as input, presented by Martínez-Barco et al. in [6] and Palomar et al. in [7], and a list of antecedents (*LA*) that appear in previous sentences. This parser returns a *slot structure (SS)* for each sentence. This *SS* stores all syntactic, lexical, morphologic and semantic knowledge as the following example shows:

The *SS* of a sentence is a feature structure formed by several arguments and named with the name itself of the constituent (*sentence, noun phrase,* etc.). The first argument, *C*, what is called *concordance knowledge*, provide us lexical, morphologic and semantic knowledge. The second one, *X*, is an identifier of the constituent (*discourse marker*). The remaining ones represent the *SS* of sub-constituents that form the main constituent. In this example, these sub-constituents are the noun and verbal phrase. Furthermore, each sub-constituent's *SS* is formed by these arguments (concordance knowledge, discourse marker and sub-constituents' *SS*).

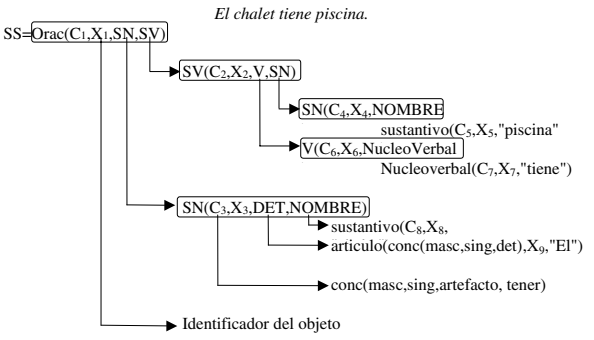


Fig. 5. *Slot Structure*

The algorithm detects definite descriptions by means of scanning the sentence's *SS* for noun phrases, i.e. arguments with name: *noun phrase*. When a noun phrase is found, then its *SS* is also scanned for a definite or demonstrative article. For example, the *SS* of the noun phrase *el chalet (the house)* has the *SS* of a definite article *the* as third argument, so it is concluded as a definite description. Otherwise, such as the noun phrase *Peter*, it is added to the *LA*. If the noun phrase is a definite description then the algorithm continues in step 4.2.

4.2 The Application of Restrictions on the List of Antecedents

Once the anaphoric expression is detected, then a set of restrictions is applied to the *LA* in order to discard antecedents that are incompatible with the definite description.

The *LA* is formed by the *SS* and a structure called *paral* for each noun phrase that has appeared in previous sentences.

The *paral* structure stores knowledge about the position of the noun phrase in the sentence.

Let us suppose the following list of antecedents, *LA*:

$$LA = [Antecedent1, Paral1, Antecedent2, Paral2, ...]$$

Each *paral* structure stores knowledge about the sentence number in which the antecedent appears, clause number, the verb of this clause, start and end word number, etc.

In order to apply restrictions, a dictionary of synonyms and relations between verbs and names is used. This dictionary has been made by hand. The following restrictions are applied:

Antecedents that do not agree in gender and number with the anaphoric expression are discarded, except those ones that have a synonymy, hyperonymy or hyponymy relation. For example, *afueras (outskirts)* and *alrededores (the surrounding area)* do not agree in gender but they are synonyms.

Antecedents that do not have the same head, synonym, hyperonym, hyponym relation nor they appear in a sentence with a verb that is related with the anaphoric expression.

If these is more than one antecedent that satisfies restrictions, then preferences are applied. Otherwise, if there is an only antecedent, then it is considered as the solution of the definite description. This antecedent is added to the *LA*. Finally, if there is no antecedent that satisfies restrictions, then the definite description is supposed to introduce a new entity in discourse and it is added to the *LA*.

4.3 The Application of Preferences on the List of Remaining Antecedents After Restrictions

If there is more than one antecedent that satisfies restrictions, then a set of preferences is applied in order to sort the antecedents after restrictions. These preferences are:

1. Antecedents with the same pre and post-modifiers are preferred.
2. The algorithm prefers those antecedents that are related with the verb of the sentence in which the definite description appears. If this relationship is direct, then antecedents with a subject syntactic function are preferred. Otherwise, if this relationship is indirect, then those ones with a complement function are preferred.
3. Indefinite noun phrases and proper nouns are preferred to definite noun phrases.
4. Antecedents that have appeared most in the text are preferred.
5. Nearest antecedent is preferred.

The scheme to apply preferences is the following. The first preference is applied to the list of antecedents that satisfies restrictions. If there is more than one antecedent that satisfies this preference, then the second preference is applied to these antecedents, i.e. the one that satisfies the first preference. And so on with the remaining preferences. This process is stopped when there is only one antecedent that satisfies a preference. In this case, this antecedent is considered as the solution of the definite description. Those preferences that are not satisfied by any antecedent are ignored, and the process continues with the following preference.

5 Evaluation

This algorithm has been evaluated on a fragment of LEXESP corpus. LEXESP corpus is formed by a set of narrative texts written by different authors. We have obtained a 95% precision[7] in classification task (anaphoric or non-anaphoric) and

[7] Precision is the quotient between number of correctly solved and number of processed definite descriptions.

an 80% precision in co-reference tasks. The kind of definite descriptions that are included in this evaluation are identity co-references produced by noun phrases in categories: *Anaphoric expressions in syntactic-semantic-textual level* (section 3).

As shown in Table 1 three experiments have been done. Experiment one consists of applying the algorithm itself. Experiment two, the algorithm has been modified erasing the restriction that discard antecedents without number and gender agreement with the anaphoric expression excepting those with a synonym, hyperonym or hyponym relationship. The final experiment consists of erasing the restriction of the antecedents without the same head or synonym, hyperonym and verb relation.

Table 1. Numbers of antecedents

Exp.	Success	Failures	Antecedents before restrictions	Antecedents after restrictions	Antecedents before preferences	Precision
1	472	24	46948	80	31	95%
2	472	24	46948	80	31	95%
3	468	28	46948	75	28	94%

We can make the same conclusion after observing table 1. Number and gender restrictions do not reduce the antecedent's amount before preferences. Therefore, it is better to use as a preference.

The main contributions of this algorithm are as follows:

We deal with co-references produced by definite descriptions that are related with a verb. These kind of anaphoric expressions are not solved by other algorithms as developed by Vieira & Poesio and Kameyama in [11] and [5], respectively.

In the first step, all definite descriptions are considered as anaphoric expressions, i.e. the algorithm tries to find out its antecedent. If the algorithm cannot find a suitable solution, then this definite description is considered as an expression that introduces a new entity. Therefore, classification mistakes do not affect the remaining anaphoric expressions as it occurs in algorithms developed by Vieria and Poesio in [11] and [10].

Antecedents of a definite description can be classified into three types: with the same head, related with a verb and with a synonym, hyperonym or hyponym relationships. We consider the same importance to each kind of antecedent. In opposition, the algorithm developed by Vieira and Poesio searches antecedents with the same head. If antecedents with the same head are not found, then the algorithm will search antecedents with a synonym, hyperonym or hyponym relation.

In works by Vieira and Poesio, [11] and [10], a 83% precision is obtained in direct anaphoras (with the same head). A 37% precision is obtained in associative anaphora (semantic relation). As Table 2 shows, our algorithm obtains better

Table 2. Results obtained applying the algorithm.

Antecedent type	Total	Failures	Successly solved	Precision
Same head	105	15	90	85.7%
Semantic relation	20	5	15	75%
Related with a verb	5	2	3	60%
Total	135	22	108	80%

results[8] : an 85.7% in anaphora with the same head and a 75% in semantic relations. Moreover, definite descriptions that are related with a verb are solved with a 60% precision.

The failures have been studied and we have found the following reasons:

Failures due to lack of semantic knowledge, since the dictionary of synonyms and relations between names and verbs (which has been made by hand) is quite small (it only has 200 entries).

The searching window used in our algorithm scopes all previous sentences. Therefore, there are a high number of possible antecedents for many definite descriptions. This fact increases the ambiguity and complexity to solve anaphora. Other algorithms such as those developed by Kameyama in [5] only use the ten previous sentences.

Some cases need more semantic knowledge to be solved. For example, when an antecedent with the same head is chosen but it is not the right solution as it has opposed modifiers, e.g. *the right ear and the left ear*.

6 Conclusion

In this paper, an algorithm to solve definite descriptions is presented. It has a precision of 80% on Spanish narrative texts. This algorithm has the following main contributions with reference to other works: it solves definite descriptions that do not have the same head as its antecedent, and do not have a synonym, hyperonym or hyponym relationships, but they have a relationships with the verb of the sentence in which the antecedent appears. Moreover, a classification of definite descriptions based on the different knowledge that is needed to solve the references.

Although other grammar categories such as the pronoun use the gender and number restriction in order to eliminate a great amount of possible antecedents, this restriction used in definite description algorithm does not eliminate great numbers of them and for this reason, it is better to use it as a preference.

Our future algorithm aims will use other semantic tools such as Spanish WordNet that will provide us with more detailed semantic knowledge. Moreover, in order to solve the problem of the best searching window (number of sentences in which the solution of an anaphora is scanned), the algorithm will scan a

[8] It should be remarked that these results cannot be directly compared since we have worked with different corpora and languages.

lower number of sentences, and if there is no solution, then a higher number of sentences will be scanned. Then, a study of the best searching window will be carried out. Furthermore, this algorithm will be applied on English texts, in order to compare it with other methods.

References

1. P. Christopherson. The Articles: A study of their theory and use in English. Copenhagen: E. Munksgaard., 1939.
2. Herbert H. Clark. Bridging. In P. Johnson-Laird and P Wason, editors, *Thinking: readings in cognitive science*, pages 411–420. Cambridge: CUP, 1977.
3. J. A. Hawkins. *Definiteness and indefiniteness.* Humanities Press, Atlantic Highlands, NJ, 1978.
4. J. R. Hobbs, D. E. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M. Tyson. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. *In E. Roche and Y. Schabes, eds. Finite State Devices for natural Language Processing. MIT Press, Cambridge, Massachussetts*, 1996.
5. M. Kameyama. Recognizing Referential Links: An Information Extraction Perspective. In R. In Mitkov and eds. B. Boguraev, editors, *Proceedings of ACL / EACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, pages 46–53, July 1997.
6. P. Martínez-Barco, J. Peral, A. Ferrández, L. Moreno, and M. Palomar. Analizador Parcial SUPP. In *Proceedings of VI biennial Iberoamerican Conference on Artificial Intelligence, IBERAMIA'98*, pages 329–341, 1998.
7. M. Palomar, A. Ferrández, L. Moreno, M. Saiz-Noeda, R. Muñoz, P. Martínez-Barco, J. Peral, and B. Navarro. A Robust Partial Parsing Strategy based on the Slot Unification Grammars. In *Proceeding of 6e Conférence annuelle sur le Traitement Automatique des Langues Naturelles. TALN'99*, pages 263–272, July 1999.
8. M. Poesio and R. Vieira. A Corpus-Based Investigation of Definite Description Use. *Computational Linguistics. MIT Press*, 24:183–216, 1998.
9. E. Prince. Toward a taxonomy of given-newinformation. *In P.Cole, editor, Radical Pragmatics. Academic Press, New York*, pages 223–256, 1981.
10. R. Vieira. Co-reference resolution of definite descriptions. In *Proceedings of VI Simposio Internacional de comunicación Social*, pages 497–503, January 1999.
11. R. Vieira and M. Poesio. *Corpus-based and computational aproach to anaphora*, chapter Processing definite descriptions in corpora. S.Botley and T. McEnery eds. UCL Press, London, 1998.

Syntactic-Conceptual Analysis of Sentences in Spanish Using a Restricted Lexicon for Disambiguation

Miguel Angel Ibarra Rivera[1], Jesús Favela Vara[1], and Aurelio López-López[2]

[1] Computer Science Department
Centro de Investigación Científica y de Educación Superior de Ensenada, CICESE
Km. 107 Carretera Tijuana-Ensenada, Ensenada, B.C., México
mairx@bahia.ens.uabc.mx, favela@cicese.mx
Telephone: (52 6) 174 50 50, Fax (52 6) 174 45 60

[2] Computer Systems Engineering Group
Instituto Nacional de Astrofísica, Optica y Electrónica
Apdo. Postal 51 y 216, 72000, Puebla, Pue., México
allopez@inaoep.mx
Telephone: (52 2) 247 20 11, Fax: (52 2) 247 25 80

Abstract. The availability of a well structured lexicon and of a parser that takes advantage of that structure for the analysis of sentences are of fundamental importance in a natural language processing system. A syntacticeonceptual analyzer of sentences in the Spanish language based on the cognitive model of Kavi Mahesh is presented. A Spanish lexicon extracted from WordNet is used for disambiguation. The terms in the lexicon are restricted to those used in the gardening domain. The structure defining each concept was widened, including for each one of the concepts three additional attributes: the actions that the entity defined by the concept usually accomplishes, the actions that others accomplish on that entity, and the actions that the entity accomplishes on itself. The analyzer itself is constituted by a finite state automata, of the augmented transition network kind, and represents a part of the Spanish grammar with which the disambiguation method is illustrated. The procedure correctly desambiguates the sentences, though is sensitive to the placement of the associated actions in the concepts within the hierarchy of the lexicon. Based on a well-structured lexicon, the method operates correctly.

Keywords: syntactic-conceptual analysis, Spanish lexicon, natural language processing, WordNet.

1 Introduction

Since the first automatic devices for information processing were built, mechanical as well as electronic, the idea of using them for the analysis of natural language texts has existed. Mainly, the purpose has been to accomplish reliable translations from one language to another, say English-Spanish, automatically, and to facilitate the man-machine interaction using natural languages.

On the other hand, by virtue of the information explosion to which we are exposed, we see an increasing need for systems that summarize documents written in natural language or support queries in daily English or Spanish. There is a need for document summaries weather on the Internet or out of it.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 538—547, 2000.

Furthermore, there are other fields that require of the natural language processing, to implement conversational agents [Allen *et al*., 1995] or support language training.

To achieve this, it is required to structure the linguistic and conceptual knowledge of the most important or necessary sections of our culture and to develop methodologies so that, interacting with these hierarchical structures or lexicons, we could automatically extract the meaning of the sentences and correctly interpret the texts. One of the best efforts of knowledge structuring is represented by WordNet, a conceptual English dictionary available now after thirty years of work [Fellbaum, 1998].

In this paper we describe a syntactic-conceptual analyzer of Spanish sentences using a WordNet subset for the disambiguation, considering terms of the activity designated as gardening.

In section 2 we present the problem of performing the syntactic-conceptual analysis of a sentence. In section 3 the syntactic-conceptual analyzer is described. It presents Mahesh's cognitive model, the syntactic analyzer based on a finite state automata, the lexicon extracted from WordNet and the syntactic-conceptual analyzer that was implemented using this lexicon.

The preliminary results obtained using this strategy are described in section 4 and, finally, in section 5 we present our conclusions and proposed future work.

2 Syntactic-Conceptual Analysis

2.1 Syntactic Analysis

Linguistic knowledge can be organized in a grammar, in the form of syntactic and semantic rules. That is to say:

Grammar: Syntax + Semantics

There are several types of grammars. Among them we find: phrase structure grammars, containing thousands of basic types of phrases; finite state grammars, that structure the linguistic patterns in finite state automata, and are complemented with statistical methods; another type of grammar consists of constraint-based formalisms. [Uszkoreit & Zaenen, 1996]. In this work we use a finite state grammar, that will be described forward in Section 2.2.

2.2 Semantic Analysis

A single word may have various meanings, depending on the context of the discourse. For example, the word "ball" can be referred to a round object, or to a lavish formal dance, between other possible meanings, such the following sentences, containing it, show:

(1) The ball traveled 90 mph on his serve
(2) She was the loveliest girl at the ball
(3) He threw nine straight balls before the manager yanked him
(4) Jim launched a ball

In the first sentence, the word "ball" represents a round object that is hit or thrown or kicked in games, while in the second it plays the roll of a lavish formal dance. In

the third sentence, it means a pitch that is not in the strike zone, as can be seen in WordNet. To disambiguate the sentence is necessary to consider the context in which those words are included in that sentence. However, in (4) there is no immediate manner of deciding. Thus, for the disambiguation to be automated, the different meanings of a word should be registered in a conceptual dictionary, known as lexicon.

2.3 Disambiguation

The main problem that exists in the analysis of a sentence is the selection of the correct meaning of that sentence. The different meanings of the words give rise to several possible meanings of the sentences. It is this variety of interpretations the one which we wish to resolve. This is known as disambiguation. There are two basic approaches to achieve the disambiguation: The rule based and the probabilistic one. In the rule based approach thousands of rules that anticipate the different situations are registered. This approach can't disambiguate everything, but when they do it, they do it well. In the probabilistic approach practically every sentence is disambiguated, but sometimes erroneously. This is by virtue of the fact that it does not leave ambiguous sentences, and upon forcing the disambiguation choosing the most probable meaning a mistake can be made.

2.4 Lexicons

A lexicon is a hierarchical structure, a tree or a set of trees, in which the nodes are words that represent concepts. In the hierarchy, the father of a node, or concept, is the category to which that concept belongs; while the children of a node, or concept, are the different types of instances of that concept.

For example, take the word "tree". In WordNet there are two nodes, in different branches, for this concept:

tree -- (a tall perennial woody plant having a main trunk and branches forming a distinct elevated crown; includes both gymnosperms and angiosperms)

tree, tree diagram -- (a figure that branches from a single root; "genealogical tree")

The location of each one of these nodes in the hierarchy is:

tree => woody plant, ligneous plant => vascular plant, tracheophyte => plant, flora, plant life => life form, organism, being, living thing => entity, something

tree, tree diagram => plane figure, two-dimensional figure => figure => shape => attribute => abstraction

Some of their children nodes are:

tree
=> yellowwood, yellowwood tree
=> lancewood, Oxandra lanceolata
=> negro pepper, Xylopia
…
=> anise tree

tree, tree diagram
=> cladogram

Recently it has been recognized the fundamental importance that the adequate construction of lexicons has [Pustejovsky, 1995]. No natural language processing project can be carried to a real system if we don't have an adequate lexicon [Guthrie *et al.*, 1996].

To develop syntactic-conceptual analyzers in Spanish language is necessary to build Spanish lexicons. Such lexicons can be built by translating WordNet subsets manually. Paradoxically, it is not possible to do this translation automatically, neither partially. The reason is that any machine translation system that attempts to translate it to the Spanish would have to rely on a syntactic-conceptual analyzer that at the same time would require a Spanish lexicon as complete at least as WordNet itself. But this is precisely the problem that we are dealing with.

We consider of great importance the development of new methodologies for the analysis of Spanish sentences based on hierarchical lexicons such as WordNet. Nowadays, the University of Amsterdam is the coordinator of the EuroWordNet Project, that includes WordNet type lexicons in Dutch, Spanish, Italian, German, French, Czech, and Estonian. The Spanish WordNet has 23,370 nodes.

3 Syntactic-Conceptual Analyzer

3.1 Mahesh's Cognitive Model

Within the natural language processing models proposed we find a promising one, by virtue of the fact that it takes into account the form in which we disambiguate the sentences: the *Cognitive Model*, of Kavi Mahesh [Mahesh, 1995]. Although we do not tie us faithfully to this model in our work, we were inspired by its general principles, and they are:

> *Principle 1. Early Selection*: The language processor selects a unique interpretation when the necessary information is available.
>
> It does not simply produce all the possible interpretations and then lets other modules select the best, but it rather considers that this selection or ambiguities resolution is an integral part of the task of understanding a sentence. Many other models violate this principle.
>
> *Principle 2. Incremental Interpretation*: The language processor produces interpretations incrementally, on line.
> Similarly as with the model HPSG [Pollard & Sag, 1994], the cognitive model bases its incremental interpretation on evidence provided by recent psycholinguistic theories. Not only it is required for the processor to choose an interpretation as soon as possible, but it must do so incrementally, for example after reading each word, instead of doing it until ends reading the sentence.
>
> *Principle 3. Integrated Processing*: The language processor must apply any type of knowledge, syntactic, semantic, conceptual, as soon as it be available.

If a piece of knowledge is available and is not applied, the processor will not be able to reduce the set of possible interpretations. This is in agreement with the psycholinguistic literature: humans process the language in an integrated way [Tanenhaus & Trueswell, 1995].

Principle 4. Functional Independence: The language processor will be able to apply any part of the knowledge independently if other types of knowledge are available or not.

This complements the integrated processing principle.

Principle 5. Determinism: The language processor will make no decision of any kind if there is no knowledge to justify it.

Thus, the processor will not make decisions at random. It will not choose randomly the interpretation of the set of possibilities, neither will choose the first interpretation of the list.

Next we describe the syntactic analysis and syntactic-conceptual analysis method proposed, which is based on Mahesh's cognitive model. One of the main differences of the method proposed with that of Mahesh reside in principle 4, related to functional independence. Explicitly, our prototype always does first a preliminary syntactic analysis that, when not definitive, continues with the conceptual analysis. On the other hand, the structure of the nodes of the lexicon that we use is different, since they contain the actions associated with the concept, as described in section 2.3.

3.2 Syntactic Analyzer

The syntactic analyzer is based on a finite automata, and corresponds to an augmented transition network. Having a whole slew of options to build it, we proceeded gradually. First we tested simple sentences, of the type: *Los jardineros cultivan unas azaleas*, that contains two noun phrases connected by a verb, and which corresponds to the graph shown in Figure 1.

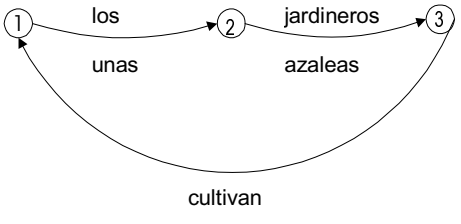


Fig. 1. The first sentences.

The sentence is constituted by the subject *los jardineros*, which is the first nominal phrase, and the predicate *cultivan unas azaleas*, itself formed by the verbal phrase *cultivan* and the second noun phrase *unas azaleas*.When the input sentences does not follow this pattern, the sentences are not recognized.

In the next stages, the sentences considered were of the style of: *Los eficientes jardineros nuevos cultivan hoy estas azaleas,* with graph shown in Figure 2.

Finally, the sentences used were of the style: *Los hijos de los jardineros eficientes van a ir a cultivar mañana unas bonitas azaleas amarilla,* or well: *Los hijos de los jardineros piensan ir a construir una bonita mesa de madera de pino de California.*

The graph that we finally came up with has less than 20 nodes, it can analyze sentences of certain complexity. Once it has been seen that the method operates well, the graph will be increased, thus expanding gradually the grammar.

In each one of the states, the decision to go to the following state does not depend only on the entry represented in the exit arch but also of the entry to the following neighboring arch; that is to say, the path followed within the graph is determined by the reading of the current grammatical category and the one that follows.

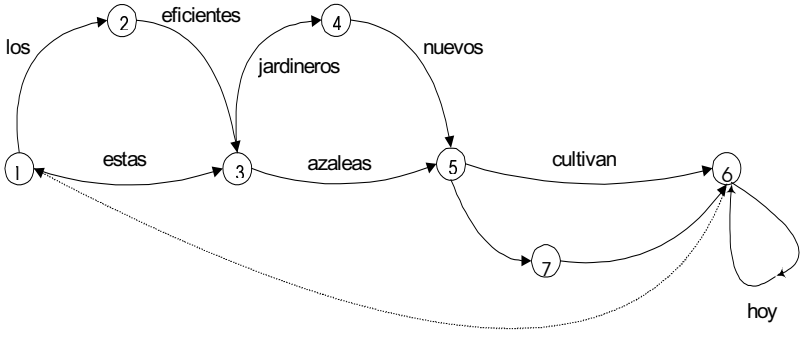


Fig. 2. Example of a more complex sentence.

3.3 The Lexicon

The lexicon was built based on the concepts that appear in a list of sentences sampled from gardening texts. Some of these sampled terms are: *abril, estanque, jardín, madera, sol, manos, fertilizante, esquinas, resequedad, jardinero*. In total, there were extracted approximately 360 terms, representing all the grammatical categories, most of them nouns. These concepts were located in WordNet from which the restricted Spanish lexicon was built, based on the structure of that lexicon. The final number of terms considered was about 2000, because some additional concepts from that lexicon were included since they were necessary to define the hierarchy in the lexicon used.

For example, in WordNet it is registered that *jardinero* is an *horticulturista*, that at the same time is an *experto*, that at the same time is a *persona*, an *organismo*, an *entidad*. For this reason, in the new lexicon we must include, in addition to the term *jardinero*, the terms that are above in the hierarchy, from *horticulturista* to *entidad* (see Figure 3). Each node in the lexicon, an entry that represents a concept in the used lexicon, is structured as follows:

Concept:
- Word
- Grammatical category
- Gender, number
- Father concept

Children concepts
Actions that the entity accomplishes
Actions that others accomplish on it
Actions that the entity accomplishes on itself

For instance, the concept *jardinero* contains the following information:

jardinero
sustantivo
masculino, singular
horticulturista
Juan, Pedro
cultivar, sembrar, podar, …

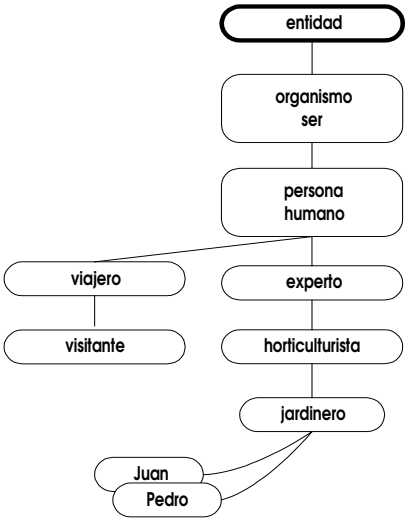


Fig. 3. The hierarchy of *jardinero*, as extracted from WordNet.

Even though the *jardinero* concept has registered specific actions that it performs, in the used lexicon, it does not have verbs associated for the actions that others fulfil on him or that he accomplishes on itself in a specific way. But as jardinero has the following hierarchy: jardinero => horticulturista => experto => persona => organismo => entidad, he inherits from *organismo* actions that he can accomplish at the same time that someone can execute on him, such as *ver*, *oir*, *oler*, and from *entidad* it inherits *dañar*, *perjudicar*, among others.

WordNet does not include information on the actions that the concept accomplishes, others accomplish on it, or it accomplishes on itself. According to Pustejovsky [Pustejovsky, 1995] , each node of a lexicon must ideally include information of the following four interpretive levels, known as the qualia structure: (1) constitutive, the relation between an object and its constituents or proper parts; (2) formal, that which distinguishes the object within a larger domain; (3) telic, purpose and function of the object; (4) agentive, factor involved in the origin or "bringing about" of an object. In our case, we try to include also information of the telic field, of Pustejovsky's qualia structure.

3.4 The Syntactic-Conceptual Parser

In the tour through the graph that represents the grammar, the analyzer tries to identify the different parts of the sentence, without taking a definitive decision on the role that each one of the input words plays. Only assumptions are made. The posible subject and predicate are identified, with all its components, and it is checked if ambiguity exists; in this case, the lexicon is consulted.

Sometimes, a simple syntactic inspection defines the subject and the predicate, as in this sentences: *Las señoras fueron a Mexicali*, or *Las azaleas están sobre la mesa*. The presence of prepositions in a sentence, in this case *a* and *sobre,* is definitive. But in sentences as *Los jardineros cultivan unas azaleas amarillas* the syntactic analysis alone can't decide if *los jardineros* is the subject or if *unas azaleas amarillas* is it. In this circumstance, the lexicon is inquired. In it, it is registered that one of the actions that *los jardineros* are accustomed to do is *cultivar*, and that one of the actions that it is regularly applied to the *azaleas*, that are plants, is *cultivar*las.

It could be that the action, the verb *cultivar*, was not registered directly in the node corresponding to the possible kernel of the subject, *jardinero*. In this case, the father node is consulted. If the action *cultivar* is not found here, we continue seeking until arriving to the root. Every action made by a term or concept associated to a node is inherited to all the subtree below it. Also, if it were registered in the lexicon that all the plants are cultivated, then the *azaleas*, being plants, also are cultivated.

Even with this restricted grammar we can already implement the disambiguation method based on the lexicon.

4 Preliminary Results

The output of the analyzer is of the following type:

Sentence to analyze:
unas plantas trepadoras hoy podará el entusiasta jardinero nuevo
SENTENCE:

PROBABLE SUBJECT 1: noun phrase [1]:
Determiner[1] (fp): *unas*
Noun[1] (fp): *plantas*
Adjective_p[1] (fp **):** *trepadoras*
VERBAL PHRASE:
Adverb[1]: *hoy*
Verb[1]: *podará*

PROBABLE SUBJECT 2: noun phrase [2]:
Determiner[2] (ms): *el*
Adjective_a[2]: adjetivo(ms**):** *entusiasta*
Noun[2] (ms): *jardinero*
Adjective_p[2] (ms**):** *nuevo*

There is gender and number concordance in the noun phrase 1.
There is gender and number concordance in the noun phrase 2.

SUBJECT: noun phrase[2]
PREDICATE: VERBAL PHRASE + noun phrase[1]

In this case, (fp) is intended for the gender and number of the word considered, that is feminine plural, while (ms) stands for masculine singular.

In case that there is not gender and number conformity in some noun phrase, a warning message is sent, but the analysis process is not stopped. On the other hand, when one of the words of the sentence is not registered in the lexicon the process is halted.

The analyzer is not always able to disambiguate the sentences. For example, in *Los jardineros nuevos prefieren las azaleas rojas* it is not possible to determine the subject since the verb *preferir* is included between the actions that execute, and are executed by, the *organismos* in general, in this lexicon.

It can be decided that only certain types of organisms could execute the action of *preferir*, in this case the persons; but this would carry us later to commit mistakes in sentences such as *Las azaleas prefieren los lugares sombreados*. One way to handle this problem is to consider the probability of use, although in this case it might still make a mistake.

This kind of examples show that the greater difficulty of building a syntactic-conceptual analyzer that disambiguates correctly is the adequate construction of the lexicon. In fact, it is not enough to translate into Spanish a subset of WordNet but one must adequately locate in the hierarchy the actions that "commonly" the instances of the concepts performs. But this task no longer corresponds to the field of computer sciences, neither to linguistics, nor to psychology. This type of tasks correspond to interdisciplinary teams of specialists as the one which built WordNet.

The execution time of the algorithm used in the analyzer depends directly on the number of words that the sentence contains and on the size of the lexicon. Basically, the time is used in the binary search within the dictionary. Using a 2,000 entries lexicon, in a personal computer, the analysis is accomplished in less than a second.

5 Conclusions and Further Work

The method proposed to achieve the syntactic analysis of Spanish sentences seems to be adequate. The grammar can be augmented easily and gradually. Also, the procedure to disambiguate using the lexicon works fine, as long as the lexicon is well designed and complete. This indicates that to reach an efficient and reliable syntactic-conceptual parser for Spanish, we have to focus on forming multi-disciplinary teams to design and build a Spanish lexicon of reasonable size,

Further work to do for the prototype of language processor for Spanish includes:

Expand the grammar, to consider more general types of noun phrases and of verbal phrases. What is immediate is a complete treatment of all types of prepositional Spanish phrases, taking into consideration the different roles that each preposition can play.

Continue reviewing and expanding the lexicon, following the qualia structure proposed by Pustejovsk, based on WordNet, as well as to systematize the constructions of lexicons in different domains. A graphical environment to easily edit lexicons will be of great help for this task.

Include morphology analysis to improve the parser, mainly in verbal expressions.

Apply the prototype of syntactic-conceptual parser on a natural language interface to interact with a system for indexing and information retrieval in digital libraries [Favela & Acosta, 1999]. This interface is planned to be used in a collaborative system intended for learning computer programming, to implement natural language generation and understanding of a virtual student.

Acknowledgments: This work was partially supported by CONACYT under grant 29729A and scholarship Id 266 provided to the first author.

References

Acosta, R., Favela, J. MIND: *An environment for the capture, indexing and retrieval of mixed-media from a digital library of graduate thesis*. First NSF/CONACYT Workshop on Digital Libraries, Albuquerque, New Mexico, Julio 7-9, 1999.

Allen, J.F., Schubert, L.K., Ferguson, G., Heeman, P., Hwang, C.H., Kato, T., Ligth, M., Martin, N.G., Miller, B.W., Poesio, M., Traum, D.R. *The TRAINS Project: A case study in building a conversational planning agent*. Journal of Experimental and Theoretical AI, 7(1995), pp. 7-48.

Fellbaum, C, (Editor). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, 1998.

Guthrie, L., Pustejovsky, J., Wilks, Y., Slator, B. M. *The Role of Lexicons in Natural Language Processing*. Communications of the ACM, Vol. 39 (1996), Num. 1 pp.63-72.

Mahesh, K., 1995. *Syntax-Semantics Interaction in Sentence Understanding*. Ph.D. Dissertation. Computer Science Departament. Georgia Institute of Technology, 1995.

Pollard, C., Sag, I. *Head-driven Phrase Structure Grammar*. Center for the Study of Language and Information Lecture Notes. Stanford University Press, 1994.

Pustejovsky, J. *The Generative Lexicon*. MIT Press, Cambridge, 1995.

Tanenhaus, M.K. and J.C. Trueswell. *Sentence comprehension*. In J. Miller and P. Eimas, eds., *Handbook of Perception and Cognition* Vol. 11: Speech and Language. New York: Academic Press. 1995.

Uszkoreit, H., Zaenen, A. *Grammar Formalisms*. Chapter 3 in *Survey of the State of the Art in Human Language Technology*. 1996. http://cslu.cse.ogi.edu/HLTsurvey/

Comparison of Conceptual Graphs

Manuel Montes-y-Gómez[1], Alexander Gelbukh[1], and Aurelio López-López[2]

[1] Center for Computing Research (CIC),
National Polytechnic Institute (IPN),
Av. Juan de Dios Bátiz, esq. Mendizabal,
Zacatenco, 07738, Mexico D.F., Mexico.
mmontesg@susu.inaoep.m
gelbukh@cic.ipn.mx
[2] INAOE, Electronics, Luis Enrique Erro No. 1
Tonantzintla, Puebla, 72840 México.
Tel. (52 22) 472011 Fax (52 22) 470517
allopez@gisc1.inaoep.mx

Abstract. In intelligent knowledge-based systems, the task of approximate matching of knowledge elements has crucial importance. We present the algorithm of comparison of knowledge elements represented with conceptual graphs. The method is based on well-known strategies of text comparison, such as Dice coefficient, with new elements introduced due to the bipartite nature of the conceptual graphs. Examples of comparison of two pieces of knowledge are presented. The method can be used in both semantic processing in natural language interfaces and for reasoning with approximate associations.

Keywords: *conceptual graphs*, *approximate matching*, *knowledge representation*.

1 Introduction*

For an intelligent knowledge-based system, it is important to be able to approximately compare two pieces of knowledge, answering the questions: How similar are the two situations? What situations in the knowledge base are similar to the given one? What pieces of knowledge could be useful for reasoning with the given one? This is similar to the behavior of a person who has just learned the piece of news that *John came late to the party*. The person recalls the similar pieces of knowledge (s)he already knows: *Last week John came late to the class*, or *Jack came to the party too*. Also, the person can generalize the available knowledge: *Boys like to attend parties*. An intelligent system should be able to model this behavior.

For this, the system should be able to compare pieces of knowledge in a quantitative manner rather than on the equal-or-not basis. The task of recalling „similar" knowledge and generalizing the available knowledge in an intelligent agent are similar to the tasks of natural language processing involving approximate matching, such as information retrieval, text mining, and abstracting. These tasks were our main motivation in this research.

* The work was done under partial support of CONACyT (including grant 32003-A), REDII-CONACyT, and CGEPI-IPN, Mexico.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 548—556, 2000.

For plain keyword set representation of text, like {*algorithm*, *binary*, *search*}, many different similarity measures are proposed, for instance, the Dice coefficient, the Jaccard coefficient, the Cosine coefficient (Rasmussen 1992), etc. For the representation with binary term weights, the Dice coefficient is calculated as follows:

$$S_{D_1,D_2} = \frac{2n(D_1 \cap D_2)}{n(D_1)+n(D_2)},$$

where $n(D_i)$ is the number of terms in D_i, and $n(D_1 \cap D_2)$ is the number of terms that the two documents D_i and D_j have in common. Because of its simplicity and normalization, we take it as the basis for the similarity measure we propose.

In this paper, we discuss an algorithm of such comparison for conceptual graphs. Conceptual graph representation incorporates the information about both the concepts involved in the situation and their relationships, e.g., [*algorithm*] → (*for*) → [*search*] → (*attr*) → [*binary*].

Conceptual graphs evolved from semantic networks. They have been used as a representation of text contents because of their expressive power close to natural language (Myaeng and López-López 1992).

In many of the conceptual graph applications, especially in the knowledge-based applications, graph matching is one of the main problems. For instance, in the field of information retrieval, different similarity measures have been described for comparing the query graph with the graphs from the knowledge base. The matching criterion most widely used for conceptual graphs is that if the query graph is completely contained in the given graph, then the given graph is relevant for (i.e., matches with) the given query graph. This criterion means that the contents of the found piece of information have to be more specific than the query piece (Huibers et. al. 1996).

A novel implementation of this criterion was proposed by Ellis and Lehmann (Ellis and Lehmann 1994). They used only the graph structure of the conceptual graphs to compare them. Their hypothesis is that for two conceptual graphs to match, their graph structure must match first. With this approach, they replace most graph matching with efficient operations on precompiled codes for graphs.

The partial matching criterion has been also used for comparing conceptual graphs. Partial matching allows the similarity between two conceptual graphs to take values between 0 and 1. Myaeng and López-López (Myaeng and López-López 1992) proposed a flexible algorithm for partial conceptual graph matching. They define the matching of two conceptual graphs as the set of all maximal common subgraphs.

Assuming we have the set of all maximal common subgraphs, we propose a flexible criterion to quantify the approximate matching expressed in the subgraphs. This criterion is based on the Dice coefficient, adapting it for our purpose.

First, we introduce the notion of the conceptual graph and describe the process of transformation of a text to a set of conceptual graphs. Then, we explain the main idea of the comparison of two conceptual graphs, and give the corresponding formulae. Finally, we give some examples of comparison of conceptual graphs.

2 Conceptual Graphs

Conceptual graphs as a way of knowledge representation were first introduced for representation of the contents of natural language texts. A conceptual graph is a network of concept nodes and relation nodes (Sowa 1983; Sowa, 1994). The concept

nodes represent entities, attributes, or events (actions); they are denoted with brackets. The relation nodes identify the kind of relationship between two concept nodes; they are denoted with parentheses.

In this paper, we suppose that the relations are of few very basic types, such as *attribute*, *subject*, *object*, etc. Thus, a phrase *John loves Mary* is represented with a graph like

$$[John] \leftarrow (subj) \leftarrow [love] \rightarrow (obj) \rightarrow [Mary]$$

and not like

$$[John] \leftarrow (love) \rightarrow [Mary].$$

The most readily available source of knowledge with complex structure is natural language text. In our experiments, to build a conceptual graph representation of a text, a morphological tagger, a syntactic parser, and a semantic analyzer are used. For example, given the phrase

Algebraic formulation of flow diagrams,

First, the morphological tagger supplies each word with a syntactic-role tag, given after the bar sign:[1]

Algebraic|JJ *formulation*|NN *of*|IN *flow*|NN *diagrams*|NNS .|.

Then a syntactic parser generates its structured representation:[2]

[[np, [n, [*formulation,* sg]], [adj, [*algebraic*]], [*of,*
[np, [n, [*diagram*, pl]], [n_pos, [np, [n, [*flow*, sg]]]]]], '.'].

The semantic analyzer generates one or more conceptual graphs out of such syntactic structure:[3]

$$[algebraic] \leftarrow (attr) \leftarrow [formulation] \rightarrow (of) \rightarrow [flow\text{-}diagram]$$

In this graph, the concept nodes represent the elements mentioned in the text, for example, nouns, verbs, adjectives, and adverbs, while the relation nodes represent some syntactic relation (including prepositions) between the concepts.

3 Comparison of Conceptual Graphs

After processing the pieces of text, we end up with sets of conceptual graphs representing their contents. From these graphs, the graph comparison process can be applied.

In general terms, our algorithm of the comparison of two conceptual graphs consists of two main parts:

1. Define the overlap of the two graphs, and
2. Measure the similarity between the two graphs as the relative size of their overlap graph.

---

[1] The tagger we use is based on the Penn Treebank tagset.

[2] The parser we use was developed by Tomek Strzalkowski of the New York University basing on The Linguist String Project (LSP) grammar designed by Naomi Sager.

[3] We do not discuss here the structure of the semantic analyser we use.

In the first step, we build the overlap graph $G_c = G_1 \cap G_2$ of the two initial conceptual graphs G_1 and G_2. This overlap consists of the following elements:

- All concept nodes that appear in both initial conceptual graphs G_1 and G_2;
- All relation nodes that appear in both G_1 and G_2 and relate the same concept nodes.

Under this definition, the overlap graph G_c is a set of all maximal common subgraphs of G_1 and G_2, and then a similar method to the one proposed by Myaeng and López-López (Myaeng and López-López 1992) can be used to build it.

An example of such an overlap is shown on Figure 1. We show the concept nodes such as [John] or [love] as the points A, B, etc., and the relation nodes such as (subj) or (obj) as arcs. In the figure, of the concept nodes A, B. C. D, E, etc., only the concepts A, B, and C belong to both graphs G_1 and G_2. Though three arcs A — B, A — C, and B — C are present between these concepts in G_1 only two of them are present in both graphs. Of course, for an arc between two common concepts to be included in G_c, it should have the same label and direction (not shown in Figure 1) in the two original graphs.

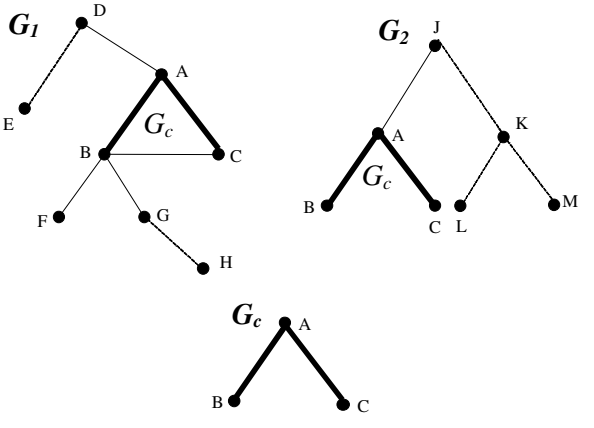


Fig. 1. Overlap of the two graphs.

In the second step, we measure the similarity between the graphs G_1 and G_2 based on their intersection graph G_c. The similarity measure is a value between 0 and 1, where 0 indicates that there is no similarity between the two pieces of knowledge, and 1 indicates that they are completely similar.

Because of the bipartite (concepts and relations) nature of the conceptual graph representations, the similarity measure is defined as a combination of two types of similarity: the conceptual similarity and the relational similarity:

The conceptual similarity measures how similar the concepts and actions mentioned in both pieces of knowledge are (e.g. topical comparison).

The relational similarity measures the degree of similarity of the information about these concepts (concept interrelations) contained in the two pieces of knowledge. That is, it indicates how similar is the context of the common concepts.

4 Similarity Measure

Given two conceptual graphs G_1 and G_2 respectively and the graph $G_1 \cap G_2 = G_c$, we define the similarity s between them as a combination of two values: their conceptual similarity s_c and their relational similarity s_r.

The conceptual similarity s_c expresses how many concepts the two graphs G_1 and G_2 have in common. We calculate it using an expression analogous to the well-known Dice coefficient used in information retrieval (Rasmussen 1992):

$$s_c = \frac{2n(G_c)}{n(G_1)+n(G_2)}$$

where $n(G)$ is the number of concept nodes of a graph G. This expression varies from 0 (when the two graphs have no concepts in common) to 1 (when the two graphs consist of the same set of concepts).

The relational similarity s_r indicates how similar the relations between the same concepts in both graphs are, that is, how similar the information about these concepts contained in the two pieces of knowledge is. In a way, it shows how similar the contexts of the common topics in both graphs are.

We define the relational similarity s_r to measure the proportion between the degree of connection of the concept nodes in G_c, on one hand, and the degree of connection of the same concept nodes in the original graphs G_1 and G_2, on the other hand. With this idea, a relation between two concept nodes conveys less information about the context of these concepts if they are highly connected in the initial graphs, and conveys more information when they are weakly connected in the initial graphs. We formalize this notion using a modified formula for the Dice coefficient:

$$s_r = \frac{2m(G_c)}{m_{G_c}(G_1)+m_{G_c}(G_2)}$$

where $m(G_c)$ is the number of the arcs (the relation nodes in the case of conceptual graphs) in the graph G_c, and $m_{G_c}(G)$ is the number of the arcs in the immediate neighborhood of the graph G_c in the graph G. The immediate neighborhood of $G_c \subseteq G$ in G consists of the arcs of G with at least one end belonging to G_c.

Figure 2 illustrates these measures. In this figure, the nodes *A*, *B*. and *C* are the conceptual nodes common for G_1 and G_2 and thus belonging to G_c. Bold lines represent the arcs (relation nodes) common to the two graphs. The arcs marked with the symbol ✓ constitute the immediate neighborhood of the graph G_c (highlighted areas), their number is expressed by the term $m_{G_c}(G_i)$ in the formula above.

The value of $m_H(G)$ for a subgraph $H \subseteq G$ in practice can be calculated as follows:

$$m_H(G) = \sum_{c \in H} deg_G c - m(H),$$

where $deg_G c$ is the degree of concept node c in the graph G, i.e., the number of the relation nodes connected to the concept node c in the graph G, and $m(H)$ is the number of relation nodes in the graph H.

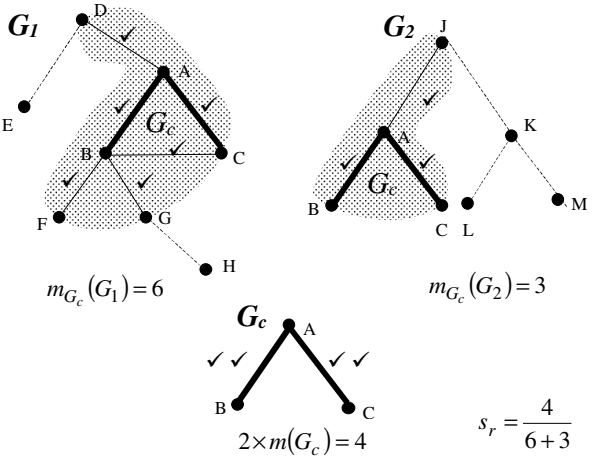


Fig. 2. Calculation of relational similarity.

Now that we have defined the two components of the similarity measure, s_c and s_r, we will combine them into a cumulative measure s. First, the combination is to be roughly multiplicative, for the cumulative measure to be roughly proportional to each of the two components. This would give the formula $s = s_c \times s_r$. However, we can note that the relational similarity has a secondary importance, because it existence depends of the existence of some common concepts nodes, and because even if no common relations exist between the common concepts of the two graphs, the corresponding pieces of knowledge are still similar to some degree. Thus, while the cumulative similarity measure is proportional to s_c, it still should not be zero when $s_r = 0$. So we smooth the effect of s_r:

$$s = s_c \times (a + b \times s_r),$$

With this definition, if no relational similarity exists between the graphs, that is, when $s_r = 0$, the general similarity only depends of the value of the conceptual similarity. In this situation, the general similarity is a fraction of the conceptual similarity, where the coefficient a indicates the value of this fraction.

The values of the coefficients a and b depend on the structure of the graphs G_1 and G_2 (i.e. their value depend on the degree of connection of the elements of G_c in the original graphs G_1 and G_2). We calculate the values of a and b as follows:

$$a = \frac{2n(G_c)}{2n(G_c) + m_{G_c}(G_1) + mG_c(G_2)}$$

where $n(G_c)$ is the number of concept nodes in G_c and $m_{G_c}(G_1) + m_{G_c}(G_2)$ is the number of relation nodes in G_1 and G_2 that are connected to the concept nodes appearing in G_c.

With this formula, when $s_r = 0$, then $s = a \times s_c$, that is, the general similarity is a fraction of the conceptual similarity, where the coefficient a indicates this portion.

Thus, the coefficient a expresses the part of information contained only in the concept nodes (according to their surrounding). It is calculated as the proportion between

the number of common concept nodes (i.e. the concept nodes of G_c) and the total number of the elements in the context of G_c (i.e., all concept nodes of G_c and all relation nodes in G_1 and G_2 connected to the concept nodes that belong to G_c).

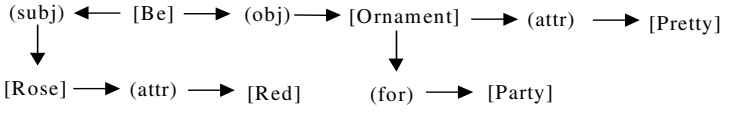
When $s_r = 1$, all information around the common concepts is identical and therefore they convey the same information in the two pieces of knowledge. In this situation, the general similarity takes it maximal similarity value $s = s_c$, and consequently $a + b \times s_r = 1$. Thus, the coefficient *b* is equal to $1 - a$, and expresses the complementary part of information conveyed in the relationships among nodes.

5 Examples

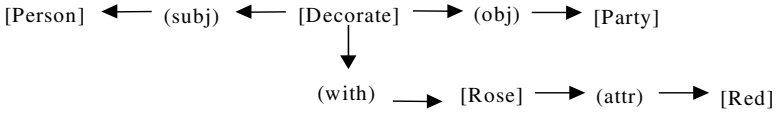
The following example shows how conceptual graphs are compared. This example consists in the comparison of three simple conceptual graphs. The relations used in the graphs are the following: *obj* (relates actions with their objects), *subj* (relates actions with their subjects), *attr* (relates concepts with their attributes) and prepositions (specific prepositions, such as *of*).

The graphs we use in our examples are the representation of simple phrases in natural language:

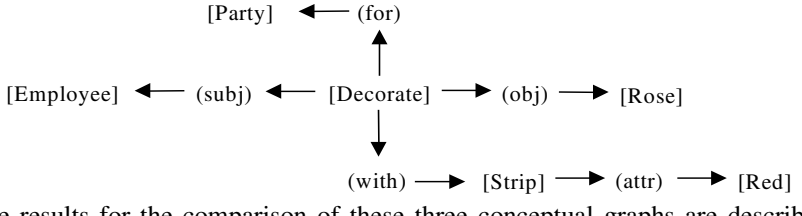
1) Red roses are a pretty ornament for a party.



2) Persons decorated their parties with red roses.



3) The employee decorated the roses with a red strip for a party.



The results for the comparison of these three conceptual graphs are described in Table 1.

Table 1. An example of comparison.

G_1 and G_2	G_c	s_c	a	s_r	s
1 and 2	[Rose]→(attr)→[Red] [Party]	0.54	0.50	0.33	0.36
2 and 3	[Rose] [Red] [Decorate] [Party]	0.72	0.47	0	0.34
1 and 3	[Rose] [Red] [Party]	0.50	0.50	0	0.25

In spite of the simplicity of the examples used, we can observe the general behavior of the measure and how the conceptual and relational similarities are combined to produce the final measure. For instance, the examples show that our measure values higher those graphs with connected common elements than the graphs with a greater number of common concepts that are not connected. This means that our similarity measure is focused on what is known *about* the concepts (interconnection of concepts) and not only on just the concepts per se.

6 Applications

Besides the direct use of the comparison technique to handle knowledge bases, our motivation points to its use in some knowledge management tasks such as information retrieval and text mining. In fact, our experiments and results have been done in these areas.

One of the main problems of current methods for information retrieval is the low precision of their results. One solution to this problem is using better representations of the text content. An example of this trend is the use of conceptual graphs as the representation of the content of texts (Myaeng 1990; Ellis and Lehmann 1994; Genest and Chein 1997).

In some of our previous work, we have also suggested using conceptual graphs in information retrieval (López-López and Myaeng 1996; Montes-y-Gómez et. al. 1999). We proposed to perform document selection by two levels of document representation. In the first level, documents are represented as keyword lists and the searching is done using traditional retrieval techniques. In the second level, documents are represents as conceptual graphs. In this level the *comparison of conceptual graphs* is done, and documents are ranked according to their similarity with the query graph. With this technique a increase in the precision is reached.

This method of comparison of conceptual graphs has also potential uses in some tasks of text mining. Currently, text mining is done at term level (Feldman et. al. 1998), and then the variety of the discovered knowledge is quite restricted.

Our main idea is to increase the potential of text mining systems again by using improved representations of text content (for instance, conceptual graphs). Thus, if texts are represented as conceptual graphs, then the comparison of those graphs emerges as a basic task. For instance, some of the text mining tasks requiring to compare text elements are: *deviation detection* (requires to compare all texts and detect the most dissimilar), *clustering* (demands to compare all texts and group those similar), and *trend discovery* (needs to compare two sets of texts and discover their differences and similarities). A way to quantify the similarities between texts is an essential element to achieve these tasks.

7 Conclusions

We have described a method for measuring the similarity between two conceptual graphs representing two pieces of knowledge in an intelligent system. The method is based on the idea of the Dice coefficient, a widely used measure of similarity for the keyword representations of texts. It also incorporates some new characteristics de-

rived from the conceptual graph structure, for instance, the combination of two complementary sources of similarity: the conceptual similarity and the relational similarity.

This measure is appropriate for comparison of pieces of knowledge since it considers not only the topical aspects (difficult to obtain from little pieces of knowledge) but also the relationships between the concepts. Thus, this approach is especially appropriate for little pieces of information organized in a semantic representation, which is the most frequent case for knowledge bases.

The method of comparison of conceptual graphs has potential uses not only in intelligent agents and knowledge bases, but also in other tasks of knowledge management, such as information retrieval systems, text mining, and document classification.

References

ELLIS G., and Lehmann F. (1994). „Exploiting the Induced Order on Type-Labeled Graphs for fast Knowledge Retrieval". Conceptual Structures: Current Practices, William m. Tepfenhart, Judith P. Dick and John F. Sowa Eds., Lecture Notes in Artificial Intelligence 835, Springer-Verlag 1994.

FELDMAN R., Fresko M., Kinar Y., Lindell Y., Liphstat O., Rajman M., Schler Y., Zamir O., (1998). „Text Mining at the Term Level". Proc. of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98), Nantes, France, September 23-26, 1998.

GENEST D., and Chein M. (1997). „An Experiment in Document Retrieval Using Conceptual Graphs". Conceptual structures: Fulfilling Peirce´s Dream. Lecture Notes in artificial Intelligence 1257. August 1997.

HUIBERS T., Ounis I. and Chevallet J. (1996). „Conceptual Graph Aboutness". Conceptual Structures: Knowledge Representation as Interlingua. Peter W. Elkland, Gerard Ellis, Graham Mann Eds. Lecture Notes in Artificial Intelligence, Springer, 1996.

LÓPEZ-LÓPEZ A., and Myaeng Sung H. (1996). „Extending the capabilities of retrieval systems by a two level representation of content". Proceedings of the 1st Australian Document Computing Symposium. 1996.

MONTES-Y-GÓMEZ M., López-López A. and Gelbukh A. (1999). „Document Title Patterns in Information Retrieval". Proc. of the Workshop on Text, Speech and Dialogue TDS'99, Plzen, Czech Republic, September 1999. Lecture Notes in Artificial Intelligence, Springer 1999.

MYAENG, Sung H. (1990). „Conceptual Graph Matching as a Plausible Inference Technique for Text Retrieval". Proc. of the 5th Conceptual Structures Workshop, held in conjunction with AAAI-90, Boston, Ma, 1990.

MYAENG, Sung H. and López-López A. (1992). „Conceptual Graph Matching: a flexible algorithm and experiments". Journal of Experimental and Theoretical Artificial Intelligence. Vol. 4, 1992, pp. 107-126.

RASMUSSEN, Edie (1992). „Clustering Algorithms". Information Retrieval: Data Structures & Algorithms. William B. Frakes and Ricardo Baeza-Yates (Eds.), Prentice Hall, 1992.

SOWA, John F. (1983). „Conceptual Structures: Information Processing in Mind and Machine". Ed. Addison-Wesley, 1983.

SOWA, John F. (1994). „Knowledge Representation: Logical, Philosophical, and Computational Foundations". Preliminary edition ICCS'94, August 1994.

Interaction of Purposeful Agents that Use Different Ontologies

Adolfo Guzmán, Jesús Olivares, Araceli Demetrio, and Carmen Domínguez

Centro de Investigación en Computación (CIC),
Instituto Politécnico Nacional (IPN). Mexico City
aguzman@cic.ipn.mx, cejaj@acm.org

Abstract. At CIC we have developed a model that enables multi-threaded agents that do not share the same ontology, to interact and interchange information among them.
The behavior of each agent is defined in a high-level language with the following features:

(1) Each agent and each interaction can be described by several sequences of instructions that can be executed concurrently. Some threads belong to an agent, others are inherited from the *scripts* which they play or perform.
(2) Of all the threads, the agent must select which ones to execute, perhaps choosing between contradictory or incompatible threads.
(3) The model allows communications between agents having different data dictionaries (ontologies), thus requiring conversion or matching among the primitives they use (§4).
(4) Some of the threads can be partially executed, thus giving rise to the idea of a "degree of satisfaction" (§6.2.1).
(5) The world on which the agents thrive suffers unexpected events (§3), to which some agents must react, throwing them out of their current behavior(s).

The model, language, executing environment and interpreter are described. Some simple examples are presented. The model will be validated using test cases based on real situations like electronic commerce, product delivery [including embedding agents in hardware], and automatic filling of databases (§6.2.2).

Keywords: *agent, ontology, unexpected events, multiple threads, incomplete execution.*

1 Introduction and Objectives

The world has now many information systems and databases, each using different data dictionaries and concept organizations.

In the field of agents, most agents are not built by *us,* but by *somebody else.* Thus, our agents will interact mostly with "unknown and never seen before" agents arriving from all over the planet. These agents will no doubt have diverse goals, and will express their wishes or purposes using different knowledge bases. Consequently, mechanisms and forms to exchange information and knowledge among heterogeneous systems are needed.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 557—573, 2000.

For versatility, an agent may have several behaviors: algorithms or threads on "how to be rich", "how to cross the street", "how to bargain", etc. An agent may be executing simultaneously several of his behaviors –those that apply–. A problem arises: some statement in thread *a* may contradict or be inconsistent with another statement in thread *b*. If so, which one is to be executed? For instance, I may be executing both the thread *traveler* of the script *on vacation* and also the thread *customer* of script *at a restaurant.* Then, I discover that the clam soup will take two hours to be cooked. "Wait that long", may say the first script, while the other may say "hurry, you may lose the plane."

An agent must obey not only his own rules (behaviors, algorithms). He must also adopt the rules imposed to him by the scripts on which he takes part. For instance, if he goes into a restaurant in the role *customer*, he must obey (most of) the rules that the script *at a restaurant* (in Marvin Minsky's sense) impose on the *customer*. He can not enter and start selling lottery tickets to other customers, although that may match with his behavior for "how to become rich." In this way, an agent acquires *additional obligations* (additional scripts to execute) on top of those with which "it was born."

Also: some scripts are incompletely executed. I do not want to eat soup. If the food becomes unpalatable, I may leave. Thus, agents may skip rules or pieces of code.

Unexpected events will happen. I was ordering more bread, when an earthquake shook the place. What to do? In general, I can not have a program to handle every possible exception (the "frame problem" of John McCarthy), but I must have more general rules. We plan to use the model of a Turing machine with two tapes, the additional one contains "unexpected events" and has an infinite number of symbols that are additional input to the (normal) behavior [Wegner, 96].

1.1 Related Previous Work; General

An agent is an asynchronous process or demon, part of a collection of similar processes, that have a common goal, and that are geographically as well as temporarily dispersed [Guzmán 98]. [Maes 94] develops agent-based systems that inherit from the user authority to act in his behalf and to make certain decisions in an autonomous and proactive (without explicit user intervention) manner. See also [Minsky 85].

In some cases, software agents are created with anthropomorphic characteristics, such as emotions [Bates 94], personality [Moon 96; Lester 97; King 96] and creativity [Boden 94]. Their purpose is to give friendly help to users and to better assist applications such as information systems.

Applications of agents vary: we find prototypes for training [Grant 97], emergent design [Edmonds 94], manufacturing [Zita 97], teaching [Selker 94; Sanchez 97; Ayala 98; Guzmán and Núñez 98], electronic commerce [Chavez 97; Noriega 97], military [Amori 92], information filters [Baclace 92], extending object based systems into agent based systems [Amandi 97; Conrad 97], agent programming without programming languages [Canfield 94], internet applications [Etzioni 94; Barret 97], virtual environments [Tu 94; Maes 95], among others.

1.1.1 Languages for Agent Execution

Declarative: KQML [Finin 93], is a message format and a message handling protocol to enable agents to interoperate and share knowledge in run time without any concern about the contents of the messages.

Imperative: TCL [Gray 97, Rus 97], is a high-level scripting language that enables a user to define mobile agents. The language is simple and imperative, and it is interpreted, in order to avoid malicious agents to cause danger in the host machine. It is embeddable in other applications and extendible with user-defined command TCL has some ways to sense hardware, software and other agents and then adapt itself, and change its behavior. Nevertheless, each define agent is not concurrent.

Tele-Script is also an important recent development in the area.

Java is multi-threaded, and is being considered as a target language of a compiler of LIA (Lenguaje de Interacción entre Agentes). We decided to create LIA, since it will allow us easier study of desired features. It differs from KQML in the sense that the former is not concerned with the content of the message and therefore makes no changes, nor equivalence among ontologies. From TCL, our proposal differs because we provide a language to emphasize the relationships among agents even in different ontologies. Both languages are not current.

1.1.2 Interaction Machines

In [Wegner 95], interaction machines are defined as extensions to the Turing Machine, to enable it to deal with incomplete specifications that can be completed interactively; it is concerned with the incompleteness of Godel's theorem. We plan to use this model, but we will organize the infinite collection of "unexpected" symbols into a finite tree of concepts, following the lines of CYC [Guha 94] and Clasitex [Guzmán 98].

1.2 Prior Research Accomplishments

ANASIN [Guzmán 94b] is a commercial product that gathers information in differed fashion, dispersed in a large region of time and space, in order to obtain strategic data, useful for decision making, from operational data sitting in files and tables located in computers installed at the work centers. The agents used in ANASIN share a single ontology or data dictionary. This described solution will be much better rendered and generalized in the work hereby proposed.

CLASITEX [Guzmán 98] is a program that finds the main topics in an article written in Spanish. It does not work on key words, but on concepts. It uses a concept tree. For this reason, it is capable of finding an article talking about shoes, even if the article does not contain such word, but it contains instead the words boot, moccasin, sandals, ..., even if these words may refer to other contexts or concepts: moccasin is also a tribe of American Indians, ...

CLASITEX++ [Beltrán *et al* 98] is a variant of Clasitex. It has a tree of concepts in English, and therefore it can analyze texts in Spanish and in English. It also has a larger tree and is written in C (Clasitex is written in UNIX shell, with sed, awk, yacc, and other utilities).

A new version of Clasitex (unnamed, by Alexander Gelbukh [Gelbukh 99, 99b], Head of the Natural Language and Text Processing Laboratory at CIC) offers several additional advantages over Clasitex. Mikhail Alexandrov, of same laboratory, is the author of a text-processing program akin to Clasitex, but with different theoretical underpinnings.

ACCESS TO UNFAMILIAR DATABASES [Guzmán 94] allowed a user familiar with the data base manager (Progress) but little familiar with the *ontology* or meaning of the key words, fields, values, file names, etc., to be able to access in a useful manner strange data bases. For instance, the user could request "give me all the graduate students studying Algebraic Topology in Germany." The system identifies the *atoms* of the user's model, and converts them (using the common sense tree described in CYC) to the corresponding atoms of the target data base, giving explanations to the user (in written Spanish) such as:

- I do not have graduate students in the database, but I have Master of Science students, Ph.D. students, and post-doctoral students.
- I do not have Algebraic Topology, but I have Mathematics.
- I do not have data about Germany, but I have about Heidelberg, Bonn, Köln, Berlin, ...

The program converts a query (posed in Spanish) into an equivalent query, in Spanish too, but where the atoms are over the ontology of the target data base.

The CYC Project [Lenat & Guha 89] tried to construct the common knowledge tree in order to solve big problems in Artificial Intelligence. A. G. worked in this project. CYC's contributions show that it is possible to form trees or taxonomies of specialized knowledge areas (which is what we intend to do here), in addition to classifying the common knowledge (goal that, due to its extension –between one and ten million concepts– was not achieved by the project). A.G. has built trees of specialized knowledge for the project "Access to unfamiliar data bases", for ANASIN (where the tree takes the form of a data dictionary) and for Clasitex.

DRAWING 3D ORGANIC MOLECULES [Olivares 95] takes as input statements of organic molecules that conform to rules of the International Union of Pure and Applied Chemistry, transforms them into a 3D graphic language, and shows them. It lets the user to apply graphical transformations (such as rotations) to enable him to analyze the structural aspects of the molecules. Given a molecule, it produces an equivalent expression, but "expressed" in another ontology.

[Olivares 91], accepts as input declarative statements that register knowledge into a semantic net. It uses imperative and interrogative statements to query the semantic net. The input statements are provided in a subset of natural language and transformed to enable the system to understand them.

The Injector of Agents [Martínez 98] places demons in remote places, using the available network. It works under a variety of network types, communication types, protocols, and it supposes an hostile or disordered environment in the host, therefore needing to send several agents to become familiar with the host's environment, so that the last agent reaching it is the agent one wishes to inject.

[Huhns 98] describes a scenario similar to the one we propose here, but with single-threaded behaviors. [Huhns 97] describes how a set of autonomous agents

cooperate to coherently management of information in environments where there are diverse information sources, that is carried out by means of a common ontology. See also other works by Huhns, Singh, Mahalingam in the references.

2 Model for Agent Interaction

Our world is closed, and agents interact only with other agents. An agent has a *purpose(s)* that he tries to reach by participating in interactions or scripts. To participate in a script, he also needs to have the necessary resources indicated by the script. He may partially reach his purposes; hence the *degree of satisfaction* (§6.2.1).

An external user defines the agents, their properties, the interactions and their properties, using the LIA language. During execution, instances of agents and interactions are created. An agent can change its own purposes.

2.1 Agents

Agents represent (or mimic) real world entities, and are born with some initial threads (behaviors), among them are a few threads to handle unexpected events. Agents may also "take" (execute) threads obtained from an script in which they participate; for instance, agent Juan Pérez may "take" the thread or role "customer" in the script "at a restaurant."

Agents are atomic and are composed of several execution threads, each corresponding to a behavior or role in a script. When an agent takes (plays, performs) a role from a script, such role should be free, that is, not assigned to another agent.

During execution, exceptions may arise (because some resource is depleted, or due to an unexpected event

Features and resources of an agent are diffident via internal variables (Figure 1).

Agent's name.
Internal variables.
Purposes (as given by internal variables).
Normal (born with) behaviors (threads). In LIA
Behaviors (threads) in LIA, acquired from scripts
Behaviors (born with and acquired) for unexpected events. In LIA

Fig. 1. Components of an agent

Purposes. An agent has one or more purposes, indicated by values of internal variables.

Threads in each agent. An agent is composed of several threads, each one specifies an action or behavior to follow. Not all threads need to be active. Example: thread "how to cross the street", thread "greeting a friend." Each thread is a LIA program.

Variables in each agent. Information representing the knowledge of an agent is shared among its threads. If a thread knows that "today is Thursday", there is no way to hide this information from its other threads, or that one of these know that "today is Friday."

Both agents and interactions possess properties that are established by giving values to global, regional, internal and local variables. States, resources, roles of agents, interactions and threads are defined with the help of these variables, as follows:

- Global variables define the state of the simulation environment, and are visible for all agents and interactions (and all their threads). Example: time of day.
- Regional variables complement the state of some agents and interactions and are visible for those that declare them. Example: where agents A and B shall meet.
- Internal variables define the state of each instance of an agent, and are visible to all its threads (and to nobody else). They are composed of a name, a value, and the time of availability. Some internal variables are used to indicate the *purposes* of the agent.
- Local variables define the state of a thread and are only available to that thread.

A time-stamp in global, regional and internal variables indicates their time of availability.

2.2 Interactions or Scripts or Theater Plays

[Riecken 94] describes agent and interaction models. In our group, research focuses on the interplay, and how two (or more) interacting agents may reach each one its purposes.

Scripts or Interactions contain roles. Each role is a thread. For instance, interaction "at a restaurant" has the following roles: customer, waiter, cashier, cook. Agents take available roles of interactions in order to try to reach their purposes. For instance, if my purpose is "to satisfy my hunger", I may take the role of "customer" at the interaction or script "at a restaurant." When an agent takes a role, the thread begins to be executed by the agent. A thread has access to all global variables, to the regional variables that the agent or the interaction possess, to internal variables of the agent, and to local variables of the thread.

As consequence of the participation of an agent in several simultaneous interactions, it may happen that at a given moment, two contradictory instructions should be executed. For instance, a thread may say "go to Chicago" and the other "go to New York." CONTRADICT, the Contradiction Finder (§6.2.1) will detect and handle the contradictions, causing for instance that a given thread be suspended, reprogrammed, modified or that other threads become active.

Unexpected events alter, too, the execution of a thread. An agent may sense some events (rain) but not others (descent in air pollution). They are handled by MEI (§3.1).

2.2.1 Threads in Each Interaction

Each interaction (Figure 2) consists of several roles that contain the requirements for their activation and the benefits that the agent obtains if he executes the role.

For a role to be assigned to an agent, the role should be free and the purpose of the agent must coincide with the benefits of the role. When purpose and benefits match, the prerequisites of the role are reviewed [observing the values of the global and internal variables of the agent] and, if the agent fulfils them, then the agent gets the role (and the role becomes "assigned") and begins executing it. Example: role "cashier" has pre-requisite "can count." Example of a prerequisite on a global variable: "time must be after 12 noon".

Roles may be similar; for instance, the script "at a restaurant" may have 17 roles of the type "customer" and two of the type "waiter."

Role One: customer	Role Two: waiter	Role Three: cook
Prerequisites or requirements	Prerequisites or requirements	Prerequisites or requirements
Benefits	Benefits	Benefits
Purposes of this thread or role.	Purposes of this thread or role.	Purposes of this thread or role.
Local variables.	Local variables.	Local variables.
Code in LIA. How to play this role.	Code in LIA. How to play this role.	Code in LIA. How to play this role.
Internal variables (relevant to the script or interaction)		

Fig. 2. Components of the interaction or script "At a restaurant"

Each instruction is scheduled for execution by inserting it in the queue with an stamped execution time. When this time arrives, all instructions with that time-stamp are executed.

2.2.2 Communication among Agents

Communication among agents has followed two approaches: (1) imperative languages, where the user specifies what the agent should do [Rus 97]; (2) declarative languages, used for information exchange, for instance, KQML [Finnin 93 and 94]. In LIA, communication is indicated by *accept* and *output* statements in each thread. For instance, role "customer" may *output* "I want some bread", which is *accepted* by an accept instruction in role "waiter". Inputs that an agent can sense are queue at its input port until consumption; unwanted inputs are consumed [accepted] but ignored

3 Handling Unexpected Events

Unexpected events are unforeseen happenings. Our model handles them through MEI, a machine of unexpected events sitting outside the agents' environment, that (1) produces the unexpected events at unexpected times, (2) find out which agents can *perceive* the unexpected events, (3) interrupts all the agent's threads, (4) decides which *reaction behavior* (if any) should be activated in response to the event, (5) reactivates some of the interrupted threads, (6) detects the end of the event, (7) may activate additional threads, and (8) usually stops the reaction behavior.

3.1 Parts of MEI (Máquina de Eventos Inesperados)

- Generator of Unexpected Events. It generates and sends unexpected events to the LIA environment, where agents interact. It generates at the beginning of time all the unexpected events (rain, earthquake, I met an old friend at a shop, ...), of the form {type of event, start time, end time, values related to the event (intensity of rain, say)}. It performs action (1) of above list.

Unexpected Event Handler. It performs actions (2)-(8), when called by LIA's interpreter.

3. 2 Handling an Infinite Number of Unexpected Events

An agent has (by birth, by acquisition from a script in which he plays a role, or by learning) a small number of canned behaviors to react to unexpected events. He may know how to react to rain (reaction depends if he is in a hurry, if he carries an umbrella, and so on), but not how to react to a volcano eruption. On the other hand, there is an infinite number of *types* of unexpected events. Being impossible for an agent to be born with (or to acquire) an infinite number of reaction behaviors, agents share the *tree of unexpected events,* where each event has a LIA thread on how to react to such event (Figure 3). This tree is infinite, but –as we said– an agent (a) can *sense* or perceive only a subset of unexpected events, and (b) possesses only a small number of reaction behaviors. Thus, • an agent reacts only to events which he can sense, and • he uses the tree to find out which is the most specific reaction behavior (to the event) that he possesses, and uses it. For instance, if an agent does not have a "volcano eruption" reaction behavior, then he may probably use the "life threatening" reaction.

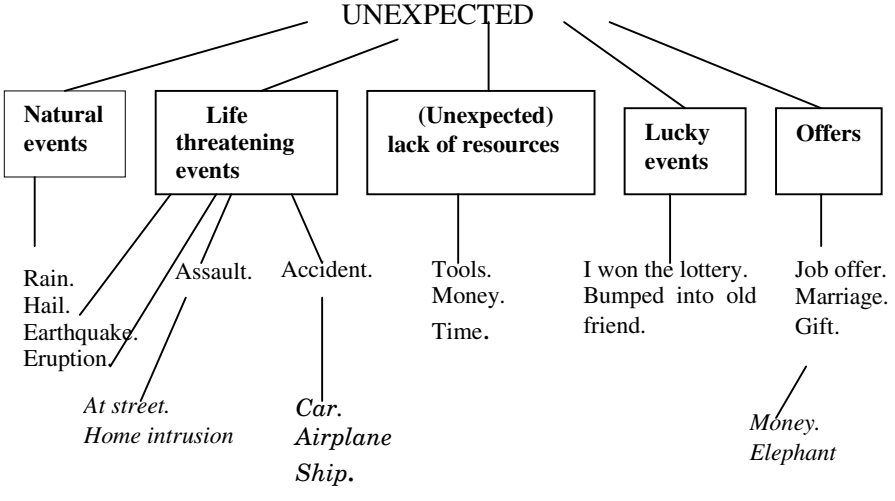


Fig. 3. Three of unexpected events.

The tree is really a lattice: a natural event (volcano eruption) can also be a life-threatening event.

Usually, a reaction behavior has parameters which define the precise reaction: Is the agent in a hurry? In a covered place? Does he carry an umbrella? Thus, depending on circumstances, an agent may decide to ignore *and not to react* to a sensed unexpected event: if he is being chased by a lion, he may ignore the rain.

4 Communication among Agents Possessing Different Ontologies

Agents written by all sorts of people must interact. For this interaction to be possible, two agents must share a common ontology (such as the common sense tree of CYC [Lenat & Guha 89]). Nevertheless, ontologies and depth of knowledge vary somewhat among agents. This section explains how to establish useful communication in spite of this.

For us, an ontology is a taxonomy (or tree) of *concepts* (not of words –thus, ontologies are language-independent). Since an agent can not transmit *concepts* to another agent,[1] he must use *words* in his preferred natural language. For our research, agents communicate through triplets of the form {entity; relationship; attributes} –notice similarity with E-R-A model of data bases– which roughly can be seen as {subject; verb; attributes}, for instance {Chevrolet car; sell; red, 1994, $5000}. To simplify our work, we assume that all agents share the same *verbs* (relationships, such as sell, buy, rent, and other verbs of electronic commerce), but the subjects may be different, such as car and automobile. We are also assuming that there are no mistakes in concept to word translation. Thus, "car" for an agent may not mean "airplane" for another agent, although "mole" for an agent may mean *the molecular weight of a substance, expressed in grams,* and for another agent it may mean *a spicy Mexican dish*, since the word mole has, in fact (as almost *any* word), several meanings or concepts: (1) a rodent; (2) a blemish of the skin; (3) a molecular weight; (4) spicy Mexican dish. As stated in [Guzmán 98], words are ambiguous; concepts are unique.

Each agent has a somewhat different ontology (a slightly different *dialect,* if they were languages). How is it possible for an agent to understand unknown words posed by the other agent? This is solved by COM, the Ontologies Matcher.

4.1 Matching Words Arising from Concepts in Different Ontologies

To explain how COM works, let us use the example that a push agent *A* issues the triplet {maize; sell; ton, $50} to a corresponding pull agent *B*. Two cases arise:

(1) *B* understands *maize,* that is, it has that word attached to one (or more) node in its ontology tree. *B* just has to ascertain the precise meaning that *A* wants to convey with *maize*. To do this, *B* goes to the father (generalisation) of *maize* in *B*'s ontology, and finds the concept *cereal,* which he transforms into the word cereal, which transmits (in the form of a question) to *A*: {questioning; cereal; meaning, father-of, maize}. *A* checks its own ontology, and finds that the father of *maize* is

[11] Once a global agreement regarding which concepts to share and how to structure the (shared, global) ontology is reached, concepts (nodes of the ontology) can be usefully transmitted.

also cereal. He answers {agreeing; cereal; meaning, father-of, maize}. Having two matches, both agents *A* and *B* are reasonably sure that they mean the same concept with *maize.* There is a match: *A* sells something that *B* most likely wants to buy. Perhaps they have to interact somewhat more, at the ontology level, to specify what color of maize *B* wants to buy, its size, etc. Perhaps *A* sells African maize, but *B* wants to buy Mexican maize.

If *A* were to issue {mole; sell; $5}, *B*'s question to *A* would be: {questioning; spicy Mexican dish, rodent, skin blemish, molecular weight of substance; father-of, mole}, to which *B* has to respond which of the four meanings is the intended one.

(2) *B* does not understand maize, so that he answers something like {questioning; maize; son-of, banana}, since *B* is an agent that buys grains and fruits, and he guesses that *A* wants to sell some kind of banana. *A* reacts as follows:

(2.a) If *A* has another word for the same concept, he issues it: {corn; sell; ton, $50}, which provokes a new call to COM from *B*.

(2.b) If *A* has no other word for the concept maize, he states that he wants to sell a particularisation of the father of maize (which is cereal): {sell; son-of, cereal; ton, $50}. To this, *B* must state that he is indeed interested in buying this kind of cereal (after all, *B* may be interested in buying heaters, not cereals). But he may or may not understand *cereal,* which may cause *A* to go to the father of cereal, *grain.*

(2.b.1) If *B* still does not understand grain, he probably is buying something else (heaters), and no communication is possible. *A* gives up.

(2.b.2) If *B* understands cereal, or grain, then basically *B* has to state that he is interested in buying this kind of grain or cereal: {buy, possibly; son-of, cereal; ton, $50}

To *B*'s answer, *A* must now *describe* this kind of cereal (or grain), namely maize, to *B*:

{sell; son-of, cereal, color, yellow, shape flat, size 1cm, ...; ton, $50}. This allows *B* to explore its own ontology, to see if *B* has some cereal with those features. Two cases:

(i) *B* has a concept matching that described by *A*, let us say, *corn.* Then *B* can decide whether he is interested in buying corn, and answers to *A* accordingly.

(ii) *B* does not have a concept matching *A*'s description. Two cases arise:

(1) *B* has a concept matching a subset of the values described by *A*. Then *B* proceeds as in case (i). For instance, *B* wants to buy a microwave oven, and *A* sells a brown microwave oven, 550 watts of power, model GE-530.

(2) *B* has no concept having a reasonable combination of features, of those explained by *A*. Then *B* gives up.

When *B* wants to match the properties of this cereal that *A* is selling, it may occur that the *description* used by *A* does not match exactly the description by *B*. For instance, *A* may use the word skin, as in color-of skin yellow, and *B* may use the word surface, or epidermis. Or, there may be a mismatch in some value: size 1 cm says *A*, while *B* understand inches; or round versus spherical . This provokes a recursive call to COM.

COM is needed because agents interact using natural words or particular concepts. The meaning of natural words is ambiguous. The meaning of a particular concept (such as good-quality-heater) is not universal. In both cases, the other agent needs to be sure which concept he is receiving. Once enough agreement is reached among grain merchants (people), as to exactly how to describe their grains, and what properties and values to use, grain agents can exchange concepts (on the *common* ontology tree agreed by the grain merchants), and ontology matchers like COM will no longer be needed. Also, the use of mark up languages, such as XML, can help in the matching of COM, since the agents can interchange the names (such as maize) and the ontology they use, this as a XML file.

5 The LIA Language

LIA (Lenguaje de Interacción entre Agentes) is used to define the simulation environment, the agents (§2.1) and the interactions (§2.2). Variables describe resources and purposes (Figure 1).
An agent is seeking to take part in interactions having roles that match his purposes. To take (play) a role, an agent must satisfy the requirements (prerequisites) established in the role.

5.1 The LIA World, Seen from Its Interpreter

During execution, instances of agents and interactions are created. Each time a role is taken, the *program counter* for that thread is initialized. As execution proceeds, each thread advances. A thread either runs to completion, or is stopped if some other thread has reached the purpose of *this* thread; or is suspended if there is resource depletion, unexpected events, conflicting threads, and in other cases. Threads can also be replanned (§6.2).

The LIA simulator or interpreter works in three stages:

A translator produces directives for the Constructor, out of LIA statements

The Constructor builds the data structures and organizes the roles so that agents can reference them. It handles main memory.

The Executor executes LIA statements, making use of MEI (§3.1), COM (§4), CONTRADICT (§6.2.1) and REPLAN (§6.2). During execution, events are displayed: creation of agents or scripts, reaching a purpose, unexpected events. Unknown LIA instructions are just displayed (printed). In this way, an instruction such as “I warmly greet agent b” gets displayed as “I, Juan Pérez, warmly greet agent b, Pedro Gómez”, which helps in the understanding and debugging of the runs. Variables referenced in LIA commands must exist, otherwise an error message is issued, but execution continues. A table contains the duration of each type of instruction; otherwise, the interpreter uses a default.

At the end of a simulation, SATIS (§6.2.1) will tell the degree of satisfaction of each agent, and of each script. Simulation occurs inside a single process (and a single computer); that is, LIA agents are not traveling or spanning the network now.

5.2 Examples

At creation of an instance of an agent, the values of its internal variables and purposes are given, either in the user command line, or inside the LIA code.
Examples.

```
CREATE AGENT PERSON Juan_Perez, Money = 1000, Hunger = YES,
                    PURPOSES
                          Hunger == NO, To_become_rich == YES
CREATE AGENT PERSON Angel_Montiel,      Money = 0,
                                        Knows_serving = YES,
                    PURPOSES
                          Looks_for_work == YES
```

In capital letters appear LIA reserved words, other tokens are variables and values. For familiarity reasons, LIA's structure resembles C.

Example of an interaction (an auction). Buyers know in general the types of auctioned products, else they can ask for samples of the type of products, to decide whether to enter. The place is a physical place described by its properties. It is a closed place whose roof may collapse on earthquakes, but not subject to flooding. Agents know the regional variables pertaining to the auctioning place, once they enter. Buyers enter the auction because they seek some product; they can review the list of auctioned products. For a buyer to enter, it is required to have money or credit. Each buyer has a *prioritized* list of products, each with a minimum and a maximum purchase price.

For each actionable product, • the auctioneer announces the product and its initial price; • each interested buyer bids; • the auctioneer select the largest bid that surpasses the previous bid (of the last round). If after some time there are no more bids, the auctioneer makes the final countdown, and assigns the product to the best available bid. In case of earthquake, the auctioneer postpones the auction. He remembers the interrupted state of the auction, to resume it later if damages are inconsequential. A buyer detecting the earthquake postpones his purchase and executes his "earthquake present" reaction behavior (he may go out, go under a desk, start to cry, ...)

During auction, COM is used to render better understanding of the traded goods. An entity (name) at an input port of agent *A* triggers a call to COM, which resolves the meaning (finds the corresponding concept in *A*'s ontology) following the procedure of §0. If an agent needs to leave, he can not continue at the auction, unless the purpose satisfied by the auction has more priority than his purpose urging him to leave.

```
global
{
int importe = 123                    //Value of article
   ,contador                  //Counter
   ,importe ;
char articulo[10][20] ;
double precio[10] ;                  //Price
int maxprod = 10;                    //Max number of products
}
agente persona                //Declaring an agent of type person
  regional                           //Regional variables
  {
      int temperatura ;              //Temperature
  }
  interna                            //Internal variables
  {
      char nombre[30] ;              //Name
      double dinero = 3000 ;  //Money
      int hambre = SI;        //Hunger = YES
      char lugar[40] ;               //Place
  }
  proposito                          //Purpose
  {
      hambre == NO;                  //Hunger == NO
      dinero > 3500;                 //Money > 3500
   }
   mei temblor()                     //MEI; earthquake
   {
       if( com(lugar ,"techado") == SI )            //If place is roofed
           lugar = "aire libre"; // go to an open (unroofed) place
   }
   mei lluvia()                      //MEI; rain
   {
        if( com(lugar ,"abierto") == SI )    //If place has no roof
            lugar = "techado"; // go to a roofed place
   }
   mei encuentradinero(int cantidad) //MEI; findsmoney (int amount)
   {
        dinero += cantidad;   //Add to my money the amount just found
    }
    mei pierdedinero(int cantidad)   //MEI; losesmoney (int amount)
    {
        dinero -= cantidad;   //Substract from my money the loss
    }
}
// DESCRIPCTION OF THE AUCTION
interaccion Subasta                  //INTERACTION; auction
{
    papel subastador(cupo 1,  //ROLE auctioneer, only one
        requisito tipopersona == "subastador",  //REQUIREMENT
        beneficio sueldo == 1000.00) //Benefits: high salary
    {
    int i ;

        // ANNOUNCES THE DEADLINE FOR REGISTERING
        finregistro = NO;            //End of registering = NO
        salida(finregistro ,RELOJ);  //Announces NOW that deadline for
                                     //registering is NO (not yet)
        tlimite = RELOJ + 01:00:00;  //Deadline is within one hour
        salida(tlimite ,RELOJ);     //Announces NOW the time of deadline
```

```
    // WAITING FOR REGISTRATION
    finregistro = SI;                    //End of registering = YES
    salida(finregistro ,RELOJ + 1:00:00);  //In an hour, it
          //announces that the deadline is YES (has arrived)
    accept(finregistro);          //Reads end_of_registering (which
   //it will be available in one hour; thus, waits for one hour

    iniciosubasta = SI;           //Auction starts
    salida(iniciosubasta ,RELOJ);
                          //He announces NOW that auction starts
    for( i = 0; i < maxprod; i++ ) //Loop to auction all products
    {
        // ANNOUNCES THE ARTICLE TO BE SOLD
        salida(articulo[i] ,RELOJ);      //Announces NOW the article
        // ANNOUNCES THE INITIAL PRICE
        salida(precio[i] ,RELOJ);
        while( mejoranprecio == SI )   //while (betterprice == YES)
        {
            // LISTEN TO THE BIDS   //We no longer show the code:
                                        // too long for this paper
            // SELECT THE HIGHEST

            // IF SELECTED BID IS BETTER THAN PREVIOS, ANNOUNCE THIS

        }
        // ASSIGN ARTICLE TO BEST BID

    }
    // END AUCTION
}
```

6 Conclusions

6.1 Evaluation of Work so Far

Work has just begun. The interpreter has been completed recently by J. O. MEI is being programmed by C. D., and COM by A. D. So far, it seems that we made a good choice in having a multi-threaded language which we can freely change. Later, we may write a compiler from LIA to JAVA. Also, the assumptions under which this work is being carried look reasonable and interesting: • agents interact in stereotyped manners called scripts; • agents communicate with mixed ontologies; • unexpected events alter the normal interactions.

6.2 Future Work

Work continues. More features will be added to LIA. Applications will begin to test the language and to suggest changes.

6.2.1 Theory

- Planning. As the result perhaps of unexpected events, current plans have to be modified or recalculated. This is handled by REPLAN.
- Detecting contradictory actions rests, among other things, on (a) detection of conflicts arising from use of common resources, such as time; (b) incompatible goals or purposes; (c) pursuing an unchanged purpose "for a while" (that is, not constantly changing the mind of an agent). J. O is constructing CONTRADICT, the Contradiction Finder, and REPLAN, the (re)planner.
- Measuring the degree of satisfaction of an agent, and of an interaction. It is performed by SATIS, a program that evaluates which purposes were reached, and to what degree.

Now, incomplete execution of a thread is possible: some threads may not run until completion, since they were stopped or suspended. Soon, some parts of a thread may be skipped (for instance, I do not want to eat soup, even if the role of "customer" in "at the restaurant" allows for soup eating. We plan to do this by marking part of the thread as optional. Also, time-outs may signal to skip secondary or less important parts of a thread.

6.2.2 Applications that We Want to Develop

- Electronic Commerce.
- Automatic programming via agents. An agent represents each code template (user interface template, data access template, ...). When presented a specification of a new program, agents look for work to do, and negotiate with each other their local interfaces.
- Automatic filling of databases. I just post what data I need; unknown agents supply me with it.
- Embedding agents into small cards. Smart articles. Distributed logistics. When my car passes the toll gate approaching Oaxaca City, its agent asks for directions to the Tecnológico de Oaxaca, to the agent in the toll gate.

Acknowledgements. Our gratitude to IPN and its director, Diódoro Guerra; to CONACYT and its director, Carlos Bazsdresch; and to REDII-CONACYT and its director, Dr. Felipe Bracho, sponsors of this project. Many ideas were suggested by or in conversations with Dr. Michael Huhns. Fruitful interaction occurred with members of the Natural Language Lab of CIC.

References

Those references marked with (•) are in Spanish.

Amandi, Analia and Price, Ana. (1997) Towards Object-Oriented Agent Programming: The Brainstorming Meta-Level Architecture. *Proc. of Autonomous Agents 97*, Marina del Rey, CA, USA

Amori, Richard D. (1992*) An Adversarial Plan Recognition System for Multi-agent Airbone Threats.* Computer Science Department, East Stroudsburg University.

Ayala, Gerardo and Yano, Yoneo. (1998) A Collaborative Learning Environment Based on Intelligent Agents. *Expert Systems with Applications* **14**, Number 1/2.

Baclace, Paul E. (1992) Competitive Agents for Information Filtering*, CACM,* No. 32.

Barret, Rob; Maglio, Paul P. and Kellem, Daniel C. (1997) WBI: A Confederation of Agents that Personalize the Web. *Proc. of Autonomous Agents 97,* Marina del Rey. CA

Bates, Joshep. (1994) The Role of Emotion in Believable Agents*. Comm. ACM*, **37**, No. 7.

Boden, Margaret A. (1994) Agents and Creativity. *Comm. ACM*, **37**, 7.

Canfield, Smith David et. al. (1994) KIDSIM: Programming Agents without a Programming Language. *Comm. ACM*, **37**, 7.

Chavez, Anthony and Maes, Pattie. (1997) *Kasbah: An Agent Marketplace for Buying and Selling Goods,* MIT Media Lab, Cambridge, MA.

Conrad, Stefan et al. (1997) Towards Agent-Oriented Specification of Information Systems. *Proc. of Autonomous Agents 97*, Marina del Rey, CA.

Etzioni, Oren and Weld Daniel. (1994) A Softbot-Based Interface to the Internet. *CACM*, **37**, 7.

Finnin, T.; Weber, J.; Widerhold, G., et al. (1993*) Specification of the KQML agent communication language* (draft). The DARPA Knowledge Sharing Initiative External Interfaces Working Group. http://www.cs.umbc.edu/kqml/kqmlspec/smecp.html.

Finin, Tim et. al. (1993b) *Specification of the KQML Agent Communication Language,* DARPA Knowledge Sharing Initiative, June 15.

Finin, Tim; Fritzon, Richard; McKay, Don and McEntire, Robin. (1994) KQML as an Agent Communication Language. *Proc. of the CIKM 94*, Gaitherburg MD, USA.

Finin, Tim et. al. (1994b) KQML as an Agent Communication Language. *CIKM 94* November, Gaitherburg, MD USA.

A. Gelbukh, G. Sidorov, and A. Guzmán. (1999) A method describing document contents through topic selection. *Workshop on String Processing and Information Retrieval,* Cancun, Mexico, September 22-24. 73-80.

A. Gelbukh, G. Sidorov, and A. Guzmán. (1999b) Document comparison with a weighted topic hierarchy. *DEXA-99, 10-th International Conference on Database and Expert System applications, Workshop on Document Analysis and Understanding for Document Databases,* Florence, Italy, August 30 to September 3. 566-570.

Grand, Stephen et. al. (1997) Creatures: Artificial Life Autonomous Software Agents for Home Entertainment. *Proc. of Autonomous Agents 97,* Marina del Rey, CA.

Gray, Robert S. (1997) Agent Tcl, in *Dr. Dobb's Journal,* March.

Guha, R.V. and Lenat, Douglas B. (1994) Enabling Agents to Work Together *CACM*, **37**, 7.

Guzmán, Adolfo. (1994) Project "Access to unfamiliar data bases". Final Report, IDASA. Mexico City. •

Guzmán, Adolfo. (1994b) Anasin. User's Manual. IDASA. Mexico City. •

Guzmán, Adolfo. (1998) Finding the main themes in a Spanish document. *Journal Expert Systems with Applications,* Vol. **14**, No. 1/2, Jan/Feb. 1998, 139-148. Handling of information in natural language (Clasitex).

Adolfo Guzmán and Gustavo Núñez. (1998) Virtual Learning Spaces in distance education; tools for the EVA Project. *Journal Expert Systems with Applications*, **15,** 34, 205-210.

Huhns, M. N. (1987) *Distributed Artificial Intelligence.* Pitman Publishing Ltd., London.

Huhns, M. N. and Bridgeland, D. M. (1991) Multiagent Truth Maintenance. *IEEE Trans. on Systems, Man, and Cybernetics,* **21**, 6, 1437-1445, December.

Huhns, M. N. and Singh, M. P. (1994) Automating Workflows for Service Order Processing: Integrating AI and Database Technologies, *IEEE Expert*, **9**, 5, 19-23, October.

Huhns, M. N.; Woelk, D. and Tomlinson, C. (1995) Uncovering the Next Generation of Active Objects, *Object Magazine*, **5**, 4, 32-40, July/August.

Huhns Michael N.; Singh, Munindar P. and Ksiezyk Tomasz. (1997) Global Information Management via Local Autonomous Agents, in *Readings in Agents,* M. N. Huhns, Munindar P. Singh, eds. Morgan Kauffmann Publishers, Inc.

Huhns, M. N. and Singh, M. P. (1997b) Internet-Based Agents: Applications and Infrastructure. *IEEE Internet Computing*, **1**, 4, 8-9, July-August.

Huhns, M. N. and Singh, M. P. (eds.) (1997c) *Readings in Agents*, Morgan Kaufmann Publishers, Inc., San Francisco, CA.

Huhns, M. N. and Singh, M. P. (1998) Managing Heterogeneous Transaction Workflows with Cooperating Agents, in *Agent Technology: Foundations, Applications and Markets,* Nicholas R. Jennings and Michael J. Wooldridge, eds. Springer-Verlag, 219-240.

King, William Josseph and Ohya, Jun. (1996) The Representation of Agents: Anthropomorphism, Agency and Intelligence. *CHI '96* Companion, Vancouver, BC, Canada

Lenat, Douglas B. and Guha, R. V. (1989) *Building large knowledge-based systems.* Reading, MA: Addison Wesley. Ontologies for common knowledge. CYC project.

Lester, James C. et. al. (1997) The Persona Effect: Affective Impact of Animated Pedagogical Agents. *CHI 97,* Atlanta, GA, USA

Mahalingam, K. and Huhns, M. N. (1997) An Ontology Tool for Distributed Information Environments. *IEEE Computer*, **30**, 6, 80-83, June.

Maes, Pattie. (1994) Agents that Reduce Work and Information Overload. *CACM*, **37**, 7.

Maes, Pattie. (1995) Artificial Life Meets Entertainment: Lifelike Autonomous Agents. *Comm. of the ACM*, November 1995, **38**, 11.

Martínez-Luna, Gilberto. (1998) *Automatic installer of systems: mobile agents with blackboard structure.* Agent injection. M. Sc. Thesis, Departamento de Ingeniería Eléctrica (Computación), Centro de Investigación y Estudios Avanzados del I. P. N. •

Minsky, Marvin. (1985) *The Society of Mind.* Simon & Schuster Inc.

Moon Youngme, Nass Clifford. (1996) Adaptive Agents and Personality Change: Complementary versus Similarity as Forms of Adaptation. *CHI '96*. Companion, Vancouver, BC.

Noriega Blanco V., Pablo C. (1997) Agent Mediated Auctions: The Fishmarket Metaphor. Memory to obtain his Ph.D., Universidad Autònoma de Barcelona, Bellaterra. Spain.

Olivares, Jesús. (1991) *Evolutive System for Knowledge Representation,* B. Sc. Thesis at I. P. N.-Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas, Mexico City. •

Olivares, Jesús. (1995) *Drawing three-dimensional molecules.* EE Dept., Cinvestav-IPN. •

Riecken, Doug. (1994) M: An Architecture of Integrated Agents. *Comm. ACM*, **37**, 7.

Rus, Daniela; Gray, Robert and Kotz, David. (1997) Transportable Information Agents. *Proc. of Autonomous Agents 1997,* Marina del Rey, CA.

Sanchez J., Alfredo; Leggett, John J. and Schnase, John L. (1997) AGS: Introducing Agents as Services Provided by Digital Libraries. *DL 97.* Philadelphia PA, USA

Selker, Ted. (1994) COACH: A Teaching Agent that Learns. *Comm. ACM*, **37**, 7.

Singh, M. P.; Huhns, M. N. and Stephens, L. M. (1993) Declarative Representations of Multiagent Systems. *IEEE Trans. on Knowledge and D. E.,* **5**, 5, 721-739, October.

Tu, Xiaoyuan and Terzopoulos, Demetri. (1994) Artificial Fishes: Physics, Locomotion, Perception, Behavior.

Wegner, Peter. (1995) *Tutorial Notes: Models and Paradigms of Interaction,* Department of Computer Science, Brown University, USA, September.

Wegner, Peter. (1996) *The Paradigm Shift from Algorithms to Interaction,* Department of Computer Science, Brown University, USA, October 14th.

Wegner, Peter and Goldin, Dina (1998a) *Mathematical Models of Interactive Computing,* Draft on Observability and Empiricism.

Wegner, Peter. (1998b) *Towards Empirical Computer Science,* Brown University, USA.

Zita, Haigh Karen and Veloso, Manuela M. (1997) High-Level Planning and Low-Level Execution: Towards a Complete Robotic Agent. *Proc of Autonomous Agents 97.*

Multi-agent Adaptive Dynamic Programming

Snehasis Mukhopadhyay and Joby Varghese

Department of Computer and Information Science,
SL280, 723W Michigan St.,
Indiana University Purdue University Indianapolis
Indianapolis, IN 46202 - 5132
Phone: (317)274 9732
Fax: (317)274-9742
{smukhopa, jvarghes}@cs.iupui.edu

Abstract. Dynamic programming offers an exact, general solution method for completely known sequential decision problems, formulated as Markov Decision Processes (MDP), with a finite number of states. Recently, there has been a great amount of interest in the adaptive version of the problem, where the task to be solved is not completely known a priori. In such a case, an agent has to acquire the necessary knowledge through learning, while simultaneously solving the optimal control or decision problem. A large variety of algorithms, variously known as Adaptive Dynamic Programming (ADP) or Reinforcement Learning (RL), has been proposed in the literature. However, almost invariably such algorithms suffer from slow convergence in terms of the number of experiments needed. In this paper we investigate how the learning speed can be considerably improved by exploiting and combining knowledge accumulated by multiple agents. These agents operate in the same task environment but follow possibly different trajectories. We discuss methods of combining the knowledge structures associated with the multiple agents and different strategies (with varying overheads) for knowledge communication between agents. Results of simulation experiments are also presented to indicate that combining multiple learning agents is a promising direction to improve learning speed. The method also performs significantly better than some of the fastest MDP learning algorithms such as the prioritized sweeping.

Keywords: adaptive dynamic programming, Markov decision process, reinforcement learning, multiple learning agents, knowledge combining

1 Introduction

A wide class of practical sequential decision-making and control problems can be formulated as Markov Decision Problems(MDP), either naturally or by means of discretization of state variables. Such problems are characterized by a transition probability matrix that depends upon the decision strategy (or policy) and a payoff matrix which determines possibly infrequent and imprecise reinforcement feedback from the environment. The objective is to determine the policy

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 574–585, 2000.

(specifying the actions to be taken as a function of the current state) that optimizes a measure of the long-term reward. Some examples of practical problems that have been investigated as MDP's include game-playing (*e.g.,* Tesauro's [6] back-gammon), and nonlinear control problems.

Bellman's dynamic programming(DP) [7] offers an exact solution method for MDP's, provided all quantities defining the MDP are completely known, and the number of states is finite. Two difficulties associated with the application of DP are: the curse of dimensionality and uncertainty about the underlying MDP. The former is an inherent difficulty of the problem domain (the number of states in MDP growing exponentially with the number of state variables), and researchers have investigated approximate methods based on function approximation to generalize over a state space. In this paper, our emphasis is only on the second difficulty which corresponds to the case where some part of the problem statement of an MDP is not known a priori.

Barto *et al*[1] discuss several approaches to the problem of real-time adaptive dynamic programming. Broadly speaking, such approaches fall into one of two categories: model-based (or indirect) and model-free (or direct) approaches. In the former, an explicit model of the environment in the form of estimated transition probabilities and pay-off matrices is maintained and updated by the learning agent. The decision-making is carried out assuming the estimated model to be the true environment (the certainty equivalence principle). In the latter class falls several reinforcement learning algorithms, *e.g.,* temporal difference and Q-learning [4]. These are characterized by a lack of explicit model and the direct estimation of the long-term values of the states, based purely on the environmental feedback. Irrespective of the approach used, the learning algorithms require a large number of experiments (state transitions) in order to achieve some measure of convergence.

In this paper, our objective is to investigate how the use of multiple agents can speed up the learning process in an adaptive dynamic programming problem. It is assumed that the agents are engaged in solving identical tasks. The only distinguishing feature is the trajectory that an agent makes in the state-space. This, in turn, implies a difference in the experiences (and, hence the knowledge structure) for the multiple agents. Consequently, the question of interest to us is whether sharing of knowledge between the agents in a population can result in increased speed of learning and improved transient learning performance. If this is the case, we will also like to investigate different strategies for information or knowledge exchange, resulting in different communication overheads, and their impact on the efficiency of learning. One motivation for such multi-agent learning comes from biological systems which rarely learn from a single source of purely personal experiences. Sharing of knowledge in a population is a natural way of augmenting one's own experiences. A related active line of research in the adaptive control literature is the use of multiple models for improved performance [5]. While the prevalent methodology is choosing one of a battery of models, our formulation combines the models of all the agents. A natural analogy in enginee-

ring systems would be the sharing of knowledge between adaptive controllers for multiple identical aircraft systems; *e.g.,* a squadron of fighter aircrafts.

With advances in computing and communication technologies a vast amount of computational power is becoming available through networking. The important question in learning is no longer the amount of computation needed in each time step, but the learning speed measured by the number of real-world experiences necessary for satisfactory performance. The computation intensive nature of adaptive dynamic programming algorithms should not discourage us from exploring the use of multiple agents with the objective of reducing the learning latency.

2 Markov Decision Problem and Adaptive Dynamic Programming

In this section we briefly review the statement of a Markov Decision Problem for the sake of completeness. This is based on the standard material found in dynamic programming or reinforcement learning literature (*e.g.,* Barto and Sutton [3]).

2.1 Markov Decision Problem

A Markov Decision Process(MDP) is defined by the following elements:

- A finite number of states of the environment $S = \{s_1, s_2, \cdots, s_n\}$.
- A finite set of actions available to an agent assumed to be same at every state: $A = \{a_1, a_2, \cdots, a_m\}$.
- A payoff function $R : S \times S \times A \rightarrow \{-1, 0, 1\}$ such that $R(s_i, s_j, a)$ is the payoff given by the environment to the agent when the latter makes a transition from s_i to s_j using action a. In this paper, this payoff(for the sake of simplicity) is assumed to be deterministic with only three possible values (where -1 corresponds to penalty, 0 corresponds to no feedback and 1 corresponds to reward).
- A state transition probability function $P : S \times S \times A \rightarrow [0, 1]$ where $P(s_i, s_j, a)$ determines the probability that the environment will make a transition to state s_j from state s_i when the agent performs an action a.

The overall utility (performance index) for the agent is $\sum_{t=0}^{\infty} \gamma^t r_t$ where r_t is the payoff received at stage t and $\gamma (0 \leq \gamma < 1)$ is a discount factor.

The objective of the agent is to determine a policy $\pi : S \rightarrow A$ which results in the maximum utility.

2.2 Dynamic Programming

As stated in the introduction, Bellman's dynamic programming provides an exact solution method for finite MDP's when the environment(including the

transition probabilities P and the payoff function R) is completely known. There are several variations of the dynamic programming algorithm. In this paper, we limit ourselves to only one popular method, called value iteration. The objective, in this case, is to determine the value $V^*(s)$ of a state s, which represents the expected long-term payoff obtained under the optimal policy when starting in state s. Once such values are determined, the optimal action in each state can be determined by maximizing the expected values of immediately future states under different actions. The basic algorithm for determining the values of states is an iterative one (value iteration) and is given by:

$$\forall i \qquad V_{k+1}(s_i) = \max_{a \in A} \left\{ \sum_{s_j \in S} P(s_i, s_j, a)[R(s_i, s_j, a) + \gamma V_k(s_j)] \right\} \tag{1}$$

Once the values of all states have converged, the optimal action $a^*(s)$ in state s is chosen as:

$$a^*(s) = arg \max_{a \in A} \left\{ \sum_{s_j \in S} P(s_i, s_j, a)[R(s_i, s_j, a) + \gamma V_k(s_j)] \right\} \tag{2}$$

In this paper, our objective is to study the adaptive version of the problem using the concept of multiple agents. The adaptive problem refers to the case when a part of the problem description is not known *a priori* so that the value iteration cannot be carried out off-line. In particular, the transition probability matrix $P(.,.,.)$ and the payoff matrix $R(.,.,.)$ are assumed unknown. In such a case, an agent has to learn the value of the states (or some related quantity) based upon its experiences encountered during on-line operation. We use an indirect method for the problem, called Adaptive Real-Time Dynamic Programming (ARTDP) by Barto *et al* [1], which is briefly described in section 3. Our objective in this paper is not to study or propose new adaptive dynamic programming methods, but merely to investigate how the learning speed can be substantially improved by using multiple learning agents.

3 Single-Agent Adaptive Dynamic Programming Algorithm

As stated in previous section our approach to single-agent real-time adaptive dynamic programming is a model-based one, using the ARTDP method proposed by Barto *et al* [1]. Following steps explain this algorithm:

– The state-transition probabilities and immediate payoff matrix are estimated based on the history or past experience of state-transitions and payoffs that an agent observes while interacting with the environment. If $n_k(s_i, s_j, a)$ represents observed number of actual transitions made from state s_i to state s_j performing an action a until time step k and if $N_k(s_i, a)$ represents the total number of times that action a was executed in state s_i, i.e., $N_k(s_i, a) =$

$\sum_{s_j \in S} n_k(s_i, s_j, a)$, then the estimated state-transition probabilities at time step k are computed as the ratio:

$$P_k(s_i, s_j, a) = \frac{n_k(s_i, s_j, a)}{N_k(s_i, a)} \tag{3}$$

- Immediate payoff, $R(s_i, s_j, a)$ can also be learned as they are observed. Since payoffs are assumed to be deterministic (for the sake of simplicity), there is no need to use a relative frequency-based approach as in the case of state-transition probabilities.
- At each time step k, the agent executes following complete value iteration algorithm using equation(1) to determine the current estimate of value $V^*(s_i)$ of a state s_i with the estimated model:
  Initialize V_k to V_{k-1}, $\forall s_i \in S$
  Repeat
    $\Delta \leftarrow 0$
    For each $s_i \in S$
      $v_k \leftarrow V_k(s)$
      $V_k(s_i) \leftarrow \max_{a \in A} \left\{ \sum_{s_j \in S} P(s_i, s_j, a)[R(s_i, s_j, a) + \gamma V_{k-1}(s_j)] \right\}$
      $\Delta \leftarrow \max(\Delta, v_k - V_k(s_i))$
  until $\Delta < \Theta$ (a small positive number)
- Once the values of all states have converged, the agent can follow a semi-greedy policy to choose an action $a(s_i)$ in current state s_i. This semi-greedy policy is implemented by assigning an execution probability to each admissible action to the current state, where probability is determined by each action's utility. At time step $k+1$, the probability that agent perform an action $a \in A(s_{k+1})$ is given according to the Boltzmann Distribution[1] as:

$$Prob(a) = \frac{e^{\frac{V_k(s_i,a)}{T}}}{\sum_{a \in A(s_i)} e^{\frac{V_k(s_i,a)}{T}}} \tag{4}$$

where $V_k(s_i, a) = \sum_{s_j \in S} P(s_i, s_j, a)[R(s_i, s_j, a) + \gamma V_k(s_j)]$ and T is a positive computational temperature parameter which is decreased according to a predetermined schedule. For example, one possibel method of choosing T_k, the temperature at instant k, is $T_k = \frac{10}{k}$.

4 Combining Knowledge of Multiple Agents

In this section, we describe how the knowledge structures of several learning agents can be effectively combined to speed up the learning process for each agent. This, together with the simulation studies presented in the following section, represents the main contribution of the paper.

The reward and the transition probability matrices constitute the knowledge for each learning agent concerning the environment. These matrices, in turn, represent a compact encoding of all the experiences of the agent. We present two

different strategies for knowledge communication between agents, termed *exhaustive* and *infrequent* communication. Under the exhaustive strategy, the agents communicate after each transition made, when their individual knowledge structures are updated to incorporate the latest experience. The knowledge structures of all agents are then combined to form a common knowledge which forms the basis for future operation for all agents. The infrequent communication, on the other hand, requires an agent to communicate its knowledge to others only when it encounters an 'unusual' state transition, defined in a suitable way. This may result in substantially less amount of communication overhead and hence, less cost. However, in simulation studies, infrequent communication is seen to result in performance that is very similar to exhaustive communication.

In the following, we describe the two forms of communication as well as the method for model combining in greater detail.

4.1 Exhaustive Communication

We propose to compute the combined transition probability using the following weighted equation

$$\bar{P}(s_i, s_j, a) = \frac{\sum_{l=1}^{N} N^l(s_i, a) P^l(s_i, s_j, a)}{\sum_{l=1}^{N} N^l(s_i, a)} = \frac{\sum_{l=1}^{N} n^l(s_i, s_j, a)}{\sum_{l=1}^{N} \sum_{s_j \in S} n^l(s_i, s_j, a)} \quad (5)$$

where $n^l(s_i, s_j, a)$ = number of transitions made by agent l from s_i to s_j under action a, and $N^l(s_i, a) = \sum_{s_j \in S} n^l(s_i, s_j, a)$ = total number of transitions made by agent l from s_i to any state under action a.

Similarly, the combined estimate of reward is computed using the equation

$$\bar{R}(s_i, s_j, a) = \frac{\sum_{l=1}^{N} n^l(s_i, s_j, a) R^l(s_i, s_j, a)}{\sum_{l=1}^{N} n^l(s_i, s_j, a)} \quad (6)$$

It can be easily seen from equations (5) and (6) that the combined transition probabilities $\bar{P}(s_i, s_j, a)$ obey the laws of probability (*i.e.*, $0 \leq \bar{P}(s_i, s_j, a) \leq 1$, and $\sum_{s_j \in S} \bar{P}(s_i, s_j, a) = 1$) and $-1 \leq \bar{R}(s_i, s_j, a) \leq 1$. It is also apparent from these equations that the agents need to communicate the matrices $n^l(s_i, s_j, a)$ instead of the probability matrix $P^l(s_i, s_j, a)$, in addition to the estimated reward matrix $R^l(s_i, s_j, a)$. However, this does not affect the amount of information that need to be communicated which is $O(n_s^2.n_a)$ where n_s = the number of states, and n_a = the number of actions.

Since the agents communicate after every transition in the exhaustive strategy, the communication overhead can be considerably reduced if memory is available to store the matrices $n^l(s_i, s_j, a)$, $s_i, s_j \in S$, $a \in A$ of all agents l, from the last transition. Since only one transition is made since last communication for an agent l, only $n^l(s_{p_l}, s_{q_l}, a_{r_l})$ has to be incremented by 1 where $(s_{p_l}, s_{q_l}, a_{r_l})$ represents the actual transition made by agent l. $n^l(s_i, s_j, a)$ for all $s_i \neq s_{p_l}$, $s_j \neq s_{q_l}$, and $a \neq a_{r_l}$ remain unchanged since the last communication.

This leads to a constant $O(1)$ information for communication for each agent. However, the memory requirement is $O(N.n_s^2.n_a)$.

Once the combined common transition probabilities and reward matrices are available, a complete value iteration can be performed until convergence with these matrices. An agent can then independently choose the action to be performed based on the values of the states, as described in the single-agent case.

Centralized Vs. Decentralized Computation for Combining Knowledge Structures: A question naturally arises as to who performs the computations necessary for combining the knowledge structures, as given by equations (5) and (6). In the centralized approach, all agents communicate their knowledge to a special *leader agent*. The leader agent combines the communicated knowledge using equations (5) and (6), performs a complete value iteration with the resulting $\bar{P}$ and $\bar{R}$ matrices, and communicates the values of the states to all agents for independent decision-making. This requires only $O(N)$ pathways for communication between the agents, but suffers from well-known disadvantages of a centralized approach such as reliability (what if the leader agent "dies"?). In a decentralized approach, all agents communicate their knowledge to every other agent. Each agent subsequently independently computes $\bar{P}$ and $\bar{R}$ matrices using equations (5) and (6), and performs independent value iterations with the resulting matrices. This approach requires all pairwise $O(N^2)$ communication pathways, and increases the amount of computation for each agent. However, such a decentralized scheme has the ability to gracefully deal with agent termination or failure.

4.2 Infrequent Communication

An interesting question in decentralized communication and control is whether it is better to communicate less information more frequently than to communicate more information less frequently. While the exhaustive communication follows the former approach (*i.e.*, communicate incrementally at every instant), the infrequent communication strategy deals with the latter approach, *i.e.*, communicate (possibly) more information less frequently. In particular, for infrequent communication, it is assumed that agents communicate their knowledge to each other (or, a team leader, as the case may be), when they encounter some 'unusual experience'. In the simulation studies reported in section 5, an unusual experience is defined as getting a non-zero feedback (either a reward or penalty) from the environment. Depending on the problem to be solved, other definitions may be considered and suitable communication strategies may be designed. For example, another notion of unusual experience that may be suitable for continuous environmental feedback problems is significant departure of short-term average reward from the long-term average reward experienced.

When an agent encounters an unusual experience, it is assumed that the agent communicates the transition probabilities and the rewards from only the initial state of the unusual transition. The objective is to communicate to other agents,

which states are unusual and the nature of the unusual experience. All other agents, subsequently, can make use of this knowledge to either avoid or prefer the given state depending on whether the unusual experience is bad or good. This results in a communication overhead of $O(n_s.n_a)$ per communication. Although this is more than the amount of communication in exhaustive communication with memory, it is worth pointing out that the frequency of such communication is much less compared to that with exhaustive communication. The infrequent communication strategy has been found to result in almost the same performance as the exhaustive strategy in simulation studies reported in section 5.

5 Simulation Results

In this section, simulation results of a state space search problem is presented, with the performance comparison of agents following different algorithms explained in sections 3, 4.1 and 4.2. We also present comparison with well-known priority-sweeping algorithm discussed by Moore and Atkeson [8].

The statespace, $-4 \leq x \leq 9$ and $-4 \leq y \leq 10$, is discretized into 182 equally spaced cells (fig 1) and posed as Markov Decision Problem, where the cells of the grid correspond to the states of the MDP. At each state two actions are possible, $u \in U = \{-1, 1\}$, which probabilistically cause the corresponding state transitions according to the following transition equation:

$$x(k+1) = 0.2x(k) + 0.6y(k) + u(k) + noise(k)$$
$$y(k+1) = 0.1x(k) + 0.5y(k) + u(k) + noise(k)$$

where $noise(k) \in [-1, 1]$ is a uniformly distributed random variable.

The operating regime of the agent, along with its discretization, is computed in such a fashion that the agent state always belongs to one of the cells. All transitions result in a reward of 0, except those that take the agent to interesting (special) states marked M and N. All transitions to states marked M result in a penalty of -1 whereas transitions to states marked N result in a reward of +1. The agent's performance is measured as the average reward agent received until a given time step.

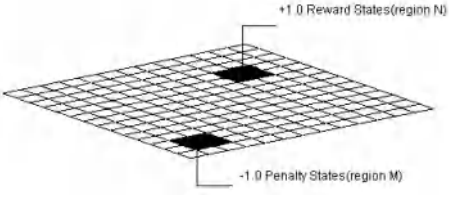


Fig. 1. Discretized statespace

The baseline experiment is based on off-line value iteration algorithm, with a discounting factor $\gamma = 0.8$. This experiment illustrates the performance of an

agent to estimate the values $V^*(s)$ of the states, given that the environment is completely known to the agent. Figure 2A shows the converged values of the states learned by an agent following off-line value iteration algorithm. The performance of such an agent as shown in figure 2B, can be used for comparison with any real time algorithm.

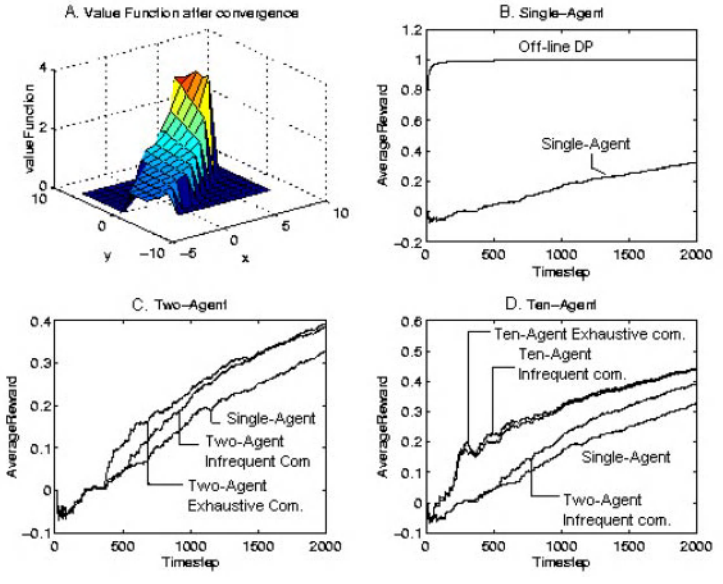


Fig. 2. A. Value Function after convergence B. Comparison: Single-Agent and off-line DP C. Comaprison: Two-Agent and Single-Agent semi-greedy approach D. Comparison: Ten-Agent and Two-Agent with infrequent communication

Different single-agent experiments were conducted, where agents followed the algorithm explained in section 3 and used the same discounting factor as in off-line experiments. These agents, varying in their random seeds, explored the same statespace by following different trajectories. Figure 2B compares learning performance of one such agent acting alone in real time with same agents' off-line learning performance. This figure also shows clearly the discernible fact that a single adaptive agent performs poorly in real-time implementations.

To unveil the advantages of using multiple agents, different experiments were carried out to explore the same statespace. These exploring agents in an agent-group, differing in their random seeds, followed different trajectories to independently estimate the values $V^*(s)$ using algorithm explained in section 3 with a discounting factor $\gamma = 0.8$.They communicate with other agents using exhaustive or infrequent strategies explained in section 4 to acquire more knowledge of the exploring statespace. As mentioned in section 4, in exhaustive communication, the agent communicates only changes to its estimate of P and R matrices after

every transition, to other agents. Agents combine this new acquired knowledge from others to generate the combined $\bar{P}$ and $\bar{R}$ matrices and uses these knowledge to choose next action. In case of infrequent communication strategy, the agent restricts its knowledge communication to those transitions which causes a change in its estimated R matrix. As an obvious consequence, these communication strategies improves the agents' model of the environment and hence improves the action chosen at following time steps. Figure 2C, demonstrates a better learning performance by an agent communicating with an another agent than the learning performance by the same agent acting alone. Figure 2D shows a further increased learning performance by the same agent acting in association with more(nine) agents. Further,Figure 2C and 2D show that agents perform better by communicating with other agents regardless of the communication strategies adopted.

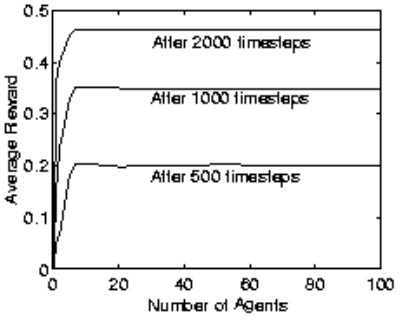


Fig. 3. Performance saturation curve

5.1 Performance Comparison with Different Number of Agents

From figure 2C and 2D, one would expect that an increase in the number of agents would improve the performance of an agent significantly. It was observed that this performance improvement is not proportional to increase in the number of agents. Instead, the performance showed a saturation level after which an increase in number of agents did not improve the agents' performance.

Different experiments were conducted with different combinations of agents, 2, 5, 7, 10, 20, 50 and 100. Figure 3 show a result from one such experimental run. It can be observed that by increasing the number of agents from 7 to 10, 20, 50 or 100 no improvement in the performance of an agent was observed by communicating with others agents.

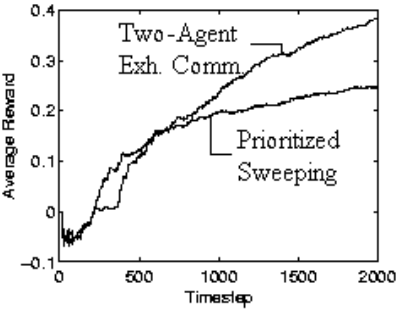


Fig. 4. Comparison: Two-Agent and Prioritized-sweeping

5.2 Comparison of Priority Sweeping Algorithm with Multi-agent Approach

Priority sweeping, proposed by Moore *et al* [8], is one of the fastest real time algorithm and can be summarized as follows:

- Each state remembers its predecessors (the states that have a nonzero transition probability to it under same action) and is assigned a priority, initialized to zero.
- At each time step k, an agent, instead of doing a complete value iteration, updates the values of m high priority states using equation(1) to determine the current estimate of value $V^*(s)$ of a state s with the estimated model as follows:
  Initialize V_{old} to $V(s)$.
  Update the state's value.
  $$V(s) \leftarrow \max_{a \in A} \left\{ \sum_{s' \in S} \widehat{P}(s, s', a)[\widehat{R}(s, s', a) + \gamma V(s')] \right\}$$
  Set the state's priority back to 0.
  Compute the value $\Delta \leftarrow |(v_{old} - V(s))|$.
  Use Δ to modify the priorities of the predecessors of s.

If after updating the value $V(s)$ at timestep k for state s, value changed by an amount Δ, then the immediate predecessors of s has its priority promoted to $\Delta \widehat{T}(s, a, s')$, unless its priority already exceeds that value. Here $\widehat{T}(s, a, s')$ represents the nonzero transition probability from state s to state s' when agent performs an action a.

Figure 4 compares the learning performance of an agent in a two-agent exhaustive communication strategy with agent's performance by adopting prioritized sweeping algorithm with 20 states being updated at each transition. Figure 4 clearly shows the advantage of communication between agents in improving the learning speed over the prioritized sweeping algorithm.

6 Comments and Conclusions

In this paper, we have demonstrated that using multiple parallel learning agents can speed up the learning process and improve transient learning performance of individual agents. Several strategies for communication and combining of knowledge of multiple agents are proposed, and their effect on learning performance is studied through simulation experiments. These experiment also reveals that, beyond a certain point, increasing the number of agents does not improve the performance further. The performance of multiple real-time adaptive agents is also compared with some of the related learning algorithms such as prioritized sweeping.

It is up to the individual designer to decide whether the improvement in performance with multiple agents is worth the extra cost in terms of computation and communication. There are many situations encountered in business and industry where, due to mass productions, several identical decision-making problems need to be solved in real-time. In such a case, one learning agent is naturally assigned to each instance of the task, and the methods proposed in this paper can be used to combine their knowledge structures. The cost of communication and the extra computational cost of combining the knowledge may be an acceptable price for the improvement of performance in this case.

Acknowledgments. Authors were supported by an NSF Career Grant 9623971 during the course of research reported in this paper. Some of the simulation experiments were performed on the NCSA supercomputing center, through a grant available through Indiana University.

References

1. A. G. Barto, S. J. Bradtke and S. P. Singh, Learning to Act using Real-Time Dynamic Programming. Artificial Intelligence, Special Volume: *Computational Research on Interaction and Agency, 72(1): 81-138*, 1995.
2. Y. Laskari, M. Metral and P. Maes, Collaborative interface agents. In *Proceedings of AAAI conference*,1994.
3. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, 1998. MIT Press.
4. L. P. Kaelbling, M. L. Littman and A. W. Moore, Reinforcement learning: A survey. In *Journal of Artificial Intelligence Research 4*, pages 237-285, 1996.
5. K. S. Narendra and J. Balakrishnan, Adaptive control using multiple models. In *IEEE Transactions on Automatic Control*, 42(2), February 1997.
6. G. Tesauro, TD-Gammon, a self-teaching backgammon program, achieves master-level play. In *Neural Computation*, 6(2), pages 215-219, 1994.
7. R. Bellman, *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.
8. A. W. Moore and C. G. Atkeson, Prioritized Sweeping: Reinforcement Learning with Less Data and Less Real Time. *Machine Learning, 13*, 1993.

A New Generation of International Databases: A Multi-agent Inspired Approach to Integrate Different Theory-Driven Databases on Conflict Warning

Monica Lagazio[1] and Evan Govender[2]

[1] University of Nottingham, School of Politics, Nottingham, NG7 2RD, UK,
Tel: 00-44-115-9514860, Fax: 00-44-115-9514859,
ldxml@nottingham.ac.uk
[2] University of Nottingham, School of Mechanical, Materials,
Manufacturing Engineering and Management, NG7 2RD, UK,
Tel: 00-44-115-9514119, Fax: 00-44-115-9514000,
epxeg1@gwmail.nottingham.ac.uk

Abstract. In many quantitative studies, international conflicts have proven difficult to explain and predict. This problem seems to have at least two related sources: 1) the attempts by political scientists to reduce the complexity of international behaviour to parsimonious models of problem-solving; and 2) the consequent lack of complementarity among international conflict databases that have been generated following these different domain definition models. Rarely are two models and their quantitative data sufficiently similar to permit either a direct comparison or an integration of their results. By combining a Neural Network approach with complexity theory, this paper puts forwards a theoretical solution for integrating different theory-driven datasets on international conflicts. An *Evolutionary Adaptive Multi-Agent Intelligent System* endorsing evolutionary strategies is suggested as a means to solve the prodigious difficulties in making use of theoretically different quantitative data. Furthermore, the multi-agent approach can provide a suitable platform to support more complex and cross-paradigmatic solutions for the study of international conflicts.

1 Introduction

The attempt of political scientists to predict international conflicts has proved to be difficult. Many quantitative studies on the subject have not yet succeeded in providing effective methodologies, which can be used to implement early warning systems for conflict prevention. The problem stems from two related sources, which involve theoretical and practical issues. From a theoretical perspective, the complexity of international behaviour has too often been reduced to parsimonious models of explanation, focusing on a small selection of explanatory variables. The construction of parsimonious problem domain models, which define the theoretical boundaries of the problem, seems unable to cope efficiently with the new type of intra-state conflicts that have characterised the Post-Cold War setting. Consequently, political scientists have come to realise that more complex problem-solving domains and cross-paradigmatic approaches are needed in order to capture the nature of international interaction. From a practical viewpoint, these different problem domain

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 586—597, 2000.

models have generated several international conflict databases, which are not functionally equivalent. Rarely are two definition models and their quantitative data sufficiently similar to permit either a direct comparison or an integration of their results. Researchers, facing prodigious difficulties in their attempts at making use of quantitative data from different theory-driven databases on conflict prevention, rather prefer to create their own customised databases. Obviously, this attitude has increased data collection costs and wasted resources as new databases are constantly created while the previous ones are no longer utilised.

The purpose of this paper is, then, to offer a theoretical solution for integrating different theory-driven databases on international conflicts. The approach suggested will make a contribution in reducing data collection cost for researchers involved in conflict prevention as well as in providing more complex and cross-paradigmatic solutions for international crises. The second and third sections of the paper put forward a neurally inspired system for the adaptation of individual databases: *the Intelligent Adaptive Information System Database.* This system produces different solution strategies in the international context by focusing on historical data provided by individual theory-driven databases. By making use of a Neural Network methodology, the system is able to create implicit theories of conflict and adapt these theories to the problem at hand. Such a system has been inspired by the main assumptions supporting the Neural Network theory, which interprets decision making as a constantly adaptive process of pattern recognition. In the fourth part of the paper the Intelligent Adaptive System Database will be further developed by adding a multi-agent perspective to it. An *Evolutionary Adaptive Multi-Agent Intelligent System*, which attempts to integrate different theory-driven databases will be suggested on the assumption that collective solutions can be generated as a result of spontaneous self-organisation. The different solutions suggested by each Intelligent Adaptive System Database on the basis of its adaptive internal model will be harmonised by a co-ordinator agent utilising evolutionary strategies.

2 The Adaptive Knowledge Cycle: The Adaptive Case-Based Reasoning Decision-Making Model

How should the international community intervene to avoid future international crises, such as Bosnia, Rwanda and Congo? A possible solution lies in using and improving existing information systems that together with computer-readable databases can provide decision-makers with predictions and recommendations on present and future crises as well as on their possible development. These information systems rely heavily on decision-making models and their capacity to offer better explanations of human events. Consequently, a redefinition of the main theoretical assumptions, which provide the rationale of the system itself, can bring about improvement in the system's performance. What we should strive for is the construction of strong theories[1] in order to build realistic models of intelligent decision-making process that can be used for the development of more efficient

[1] By strong theory it is meant those theoretical frameworks that are able to capture cognitive processes at work in political settings as realistically as possible. The term has originated from the tradition of scientific realism, which stresses the need for scientific inquiry to focus on processes operating in the world [2].

information systems. New opportunities in this direction have been opened up by major developments in cognitive science, recognising that the biological substrates of the brain play a fundamental role in the cognitive processes. The dichotomy between the mind and brain has to some degree been reconciled by experimental results that have succeeded in correlating mental phenomena, such as perception, learning, and reasoning, with patterns of nerve impulses in the brain [1]. The brain is to be regarded as a massively distributed dynamic system which, once supplied with an initial sensorial input, spreads activation impulses among its units [3]. These activation impulses are the way in which brain represents, processes, and transforms external data.

How does the cognitive process develop in the brain? Indeed, the brain works as a device that tries to make sense of all the variegated sensory/physical characteristics that comprise the input. Repetitive experiences of inputs of similar sensory nature *tune* the brain to appreciate the constant pattern, or overall relation, that exists among all the sensory features comprising the phenomenon itself.[2] Each time a new input of the same sensorial type is experienced by the agent, the brain recomputes the relationship that exists among the sensory features of the phenomena and memorises the discovered pattern via its synaptic weights. During the process of learning the external world through subsequent experiences of it, the brain keeps modifying the weight values of the synaptic connections linking the neurons that have been activated by the incoming stimulus. The knowledge that has been acquired by the system '…consists of nothing more than a carefully orchestrated set of connection weights,' while its adaptive strategy is equivalent to the value adjustment in the weights [4]. It can been said that the brain has created a specific point in his individual synaptic weight space that embeds his theory about the phenomenon. Moreover it is ready to adapt the theory constructed by subsequent learning focusing on pattern recognition strategies.

By focusing on the brain's functionality and its attribute of being a fully distributed and self-organising dynamic system, the Neural Network theory suggests a new model of decision-making. The model redefines a few important assumptions on the nature of knowledge and reasoning. First, it emphasises that knowledge and reasoning are grounded in sensory experience. It is the detection of the sensory microfeatures of the external stimulus that initiates the process of reasoning. Second, it suggests that knowledge is a constant process of adaptive learning, which is empirically driven. The cycle of experiencing the world and identifying its meanings is endless. Even if an agent starts experiencing the external environment with a set of predefined weights, which can be thought of as being selected by his evolutionary history,[3] present and future experiences would still refine the values of his synaptic

---

[2] Neural Network systems can be thought of, in the learning phase, as correlation detectors. During the learning phase the system's role is to capture higher-order statistical relationships in the input corpus, which are not explicitly represented in the sensory make up of the inputs themselves. This is accomplished if significant underlying structures can be found which have a strong correlation with the sensory structure of the inputs [6].

[3] The neurocomputational approach integrates the seemly opposite positions of empirism and nativism on the role of concepts and experiences in knowledge processing. 'Like good nativists, we have given the organism a starting point that has been selected by his evolutionary history but quite differently from them we have not fixed this structure in a rigid predeterminism. The organism can still modify these behavioural predispositions. At the same time the best of the empiricist view has been considered; there are no a priori limitations on how the organism may adapt to its environment but without endorsing the weak point of tabula rasa' [7].

connections and, consequently, alter his theory about the world. Concepts are not fixed entities stored in the agent's memory and retrieved as needed, but are constantly tuned during new interactions with the outside environment. Consequently, knowledge is part and parcel of the information process itself [5]. Finally, another conclusion that can be drawn is that decision-making is triggered by pattern recognition. By recognising constant patterns in the data corpus, agents can construct internal representations of external events, create prototypical categories of events and build up their own theories about the outside world. The agent follows, in his search for knowledge, an adaptive process, which is triggered by experiences and shaped by pattern recognition strategies. This process has been called the *Adaptive Knowledge Cycle* (Fig. 2).

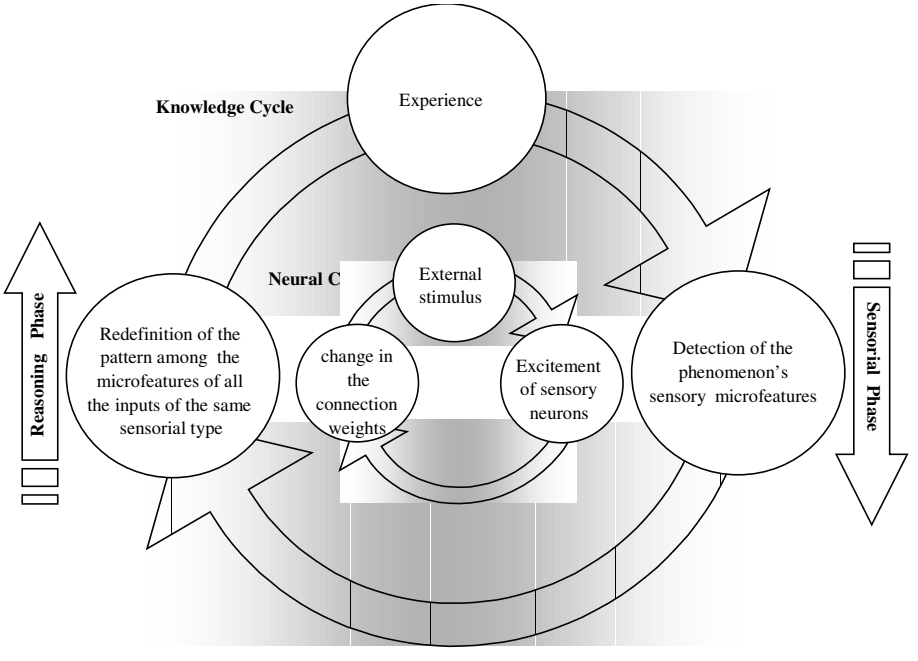


Fig. 2 The Adaptive Knowledge Cycle

3 The Intelligent Adaptive Information System Database: An Early Warning Decision Support System

The adaptive nature of the cognitive process underlined by the Neural Network approach offers an alternative for the creation of a new type of early warning decision support system for the prediction of, and intervention in international conflicts. This system (which has been named the *Intelligent Adaptive Information System Database*) will be developed following the assumption that reasoning and searching for solutions on present international problems is an inductive process directed at examining past experiences, drawing analogies, reshaping patterns in the data corpus and adapting

past solutions to newly experienced problems.[4] In order to provide the system with an adaptive capacity, it is appropriate to split the overall knowledge structure of the Intelligent Adaptive Information System Database in two phases: an *initial domain knowledge model,* and *knowledge adaptation*.

In the initial domain knowledge phase, an expert's knowledge is used to define the problem domain and then construct a proto-model for early warning in international conflicts. The initial domain knowledge modelling constitutes the starting cognitive environment where the problem is initially defined and structured. This knowledge provides both the theoretical boundaries of the problem domain, such as the main variables and their indicators, and an initial proto-theory of conflicts defining the relationship among these dimensions. The result of this phase is the construction of a theory-driven database following the theoretical boundaries decided by the expert and an initial theory that is able to explain, predict and manage international conflicts. The data included in the database is nothing more than the identified early warning dimensions of the general class of international conflicts that the expert has constructed in his brain/mind through repetitive experiences of similar situations.[5] These features will code: 1) the early warning dimensions of a historically constructed population of conflict; 2) the actions endorsed by the international community to manage the past conflicts as revealed by the expert's historical knowledge; and 3) the performance achieved by the management agent in past conflict instances, also in this case, as suggested by the historical investigation.

The knowledge adaptation phase starts when the initial domain knowledge modelling ends. Its objective is to refine the expert's initial proto-theory on the basis of new information added to the database. The adaptation is achieved by using a neurally inspired system interfaced to the theory-driven database. Fed with historical information and with data coming from new conflict events, the neural model will be able to adapt the expert's initial theory and generate a modified model for early warning in international conflicts. The adaptation of the expert's proto-theory is achieved in two subsequent steps: at the initial moment when new data related to presently developing conflicts are entered into the database, and at the end of the process when information on the outcome performance is available. This data can also be used to update the database. A computer internal model, which will refine the expert's proto-theory, is produced each time new coded data is added to the theory-driven database.

Figure 3 illustrates these two phases of the system's knowledge structure. The upper rectangular block diagrammatically represents the initial domain knowledge

---

[4] Neural Networks can be quite similar to case-based reasoning systems. As both approaches emphasise the importance of what has been called precedent logic in shaping human decision making process, we can consider Neural Networks as a development of previously established case-based reasoning algorithms. But there is a difference between the two. The difference lies in the stronger adaptive nature of Neural Network systems. For an example of how case-based reasoning methodologies have been applied to international relations researches see: Alker, James Bennett, and Dwain Mefford, Akihiko [8].

[5] The importance of precedent-based analogical reasoning in shaping human decision making has been recognised by Psychology. As Lee Roy Beach stresses, 'A decision-maker armed with all the decision technology in the world would be completely stalled if he or she could not call on past experience and knowledge about ongoing events. In this sense, recognition is implicit in any theory of decision making' [9].

model, while the lower one explains the adaptive phase. As shown by the diagram, the initial domain knowledge and the adaptive phase both share three components of the overall knowledge process.

The knowledge process typically starts when a new potential conflict needs to be processed by the system. By applying the initial domain knowledge, the early warning dimensions of the new event are coded and included in the database. The neural component will then interrogate the database and retrieve past information on the dimensions of past conflict situations, previous solutions that were attempted by the

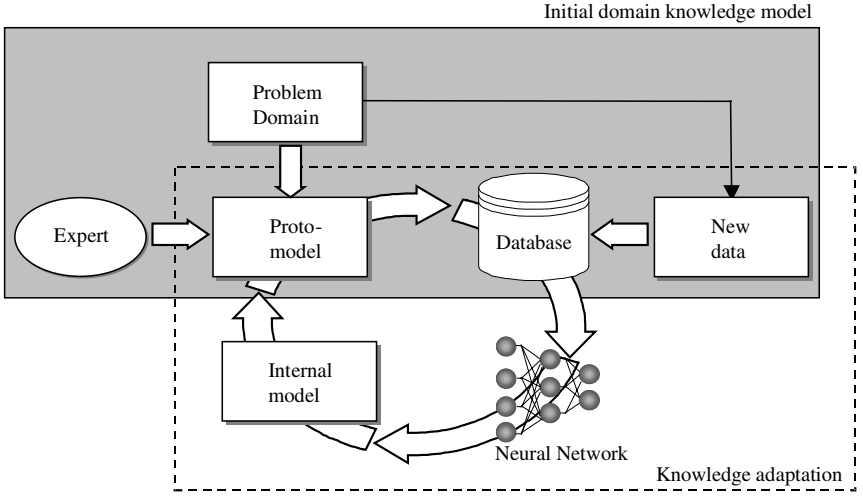


Fig. 3 The Intelligent Adaptive Information System Database

management agent, and previous outcome performances that were produced by the solutions endorsed by the intervening international actor [10]. The recommendation suggested by the system is the result of the final pattern that the neural component has constructed by computing the relation among all these inputs: information of the problem at stake which has been added to the database, information on the attempted solutions, and information on the achieved results. Understanding the problem, building up a strategy for action, and identifying the goals of the action are accomplished simultaneously by the suggested information system.[6] In a single computational effort the neurally based information system is able to construct an understanding of the phenomena, in our case a conflict event, translate this understanding into appropriate actions in response to the apprehended situation, and define the management agent's task.[7] Through this learning process based on pattern recognition a new internal model on conflict has been produced by the system as a consequence of the incoming data. This model, which represents the adaptation of the initial proto-theory created by the expert, provides the adapted solution for the new developing conflict.

---

[6] The process described above has significant similarities with the experiential learning model for institutions suggested by March and Olsen [11].

[7] Clark's example is quite clear in explaining this process of understanding and acting in the external world: 'We do not first represent the presence of a chair and then compute or infer its suitability for sitting; instead, to perceive a chair is, at one and the same time, to detect the opportunity of sitting' [12].

However, the system 's knowledge adaptation phase is not complete until the information on the outcome, becomes known. In order to complete the adaptation process the data on the solution performance need to be entered into the database. This new information provides the system with a feedback mechanism. Once again the Neural Network algorithm retrieves past data from the database and then recalculates the pattern among the early warning characteristics of the past events, solutions that were attempted in the past, the performances that were achieved, and the present performance. Consequently, a new internal computer model is generated.

4 The Evolutionary Adaptive Multi-agent Intelligent System

The Intelligent Adaptive Information System Database described in the previous section can only explain, predict, and manage conflicts within the problem domain constructed by the expert. Even if the system can adapt its proto-model, it can only modify it within the theoretical barriers initially defined for the domain. The system cannot change its unit of analysis, the early warning dimensions of the conflict or the measurement in the database. The adaptation process has an impact only in recognising the pattern among these previously determined theoretical units. For this reason, political scientists have produced various databases on conflict prevention that are shaped by different theoretical constrains. Once a new problem domain model has established itself in the literature, a database supporting these new concepts is created. However, the proliferation of theory-driven databases has created more problems than benefits [13]. From a theoretical perspective, the creation of parsimonious problem domain models seems to be unable to cope with the Post-Cold War conflicts which appear to be a much more complex event than the Pre-Cold War crises. Since the 1970s, there has been an increase in internal conflicts and quarrels among non-internationally recognised actors. These new events are challenging conceptual, methodological, and paradigmatic frameworks put forward by previous researchers. Moreover, from a practical perspective, the lack of complementarity among theory-driven databases has increased data collection costs and wasted resources as new databases are constantly created while previous ones are no longer utilised.

One way to solve the theoretical and practical issues, which have been stressed above, is to endorse a distributed intelligence framework. The more the complexity of international issues is increased by the amount of international actors interacting concurrently on a global scale, the more the search for solutions becomes a cross-paradigmatic effort. No domain model alone may formulate an efficient problem-solving environment for the new challenges facing international security. As Sherman suggests '…the criteria for establishing the dataset should be as sensitive as possible to various ideological and epistemological orientations of […] other researchers and studies' [14]. Consequently, a move towards an early warning information system that takes into account truly mixed-initiative decision-making is needed. This system should make use of aggregate adaptive intelligence and resources to cope with complex destabilising events.

By combining the Neural Network approach and complexity theory, an *Evolutionary Adaptive Multi-Agent Intelligent System,* characterised by features such as distribution, heterogeneity, adaptation, co-operation and evolution, is suggested in the following section. Such a system can be regarded as a policy recommendation tool for distributed and cross-paradigmatic decision-making environments, while offering

a model of collective decision making based on the evolutionary dynamics of complex systems. The description of the multi-agent information system has been developed in two subsequent sections. Firstly, the complex adaptive system approach, which provides the theoretical framework for the Evolutionary Adaptive Multi-Agent Intelligent System, will be outlined; secondly, a co-ordinator agent will be proposed as a means to create collective solutions from the individual recommendations suggested by each Intelligent Adaptive Information System Database. The muti-agent system's collective solutions are the product of the evolutionary mechanism that the co-ordinator agent utilises in order to adapt and organise the overall system itself.

4. 1 The Complex Adaptive Framework: Main Assumptions

Originally, a theoretical framework for the study of physical systems, the complex adaptive system approach has been successfully applied to different fields such as chemistry, biology and economics to investigate emergent macrophenomena.[8] The results achieved in these disciplines have attracted the attention of a few political scientists, who have started employing this new theoretical position to political settings by creating agent-based models *growing* social and political systems through simulation.[9] All these researches have a common commitment to the study of individual units and their interactions as a means to explain systemic structures. Indeed, the main assumption of this new analytical perspective is that large-scale events can be studied as the consequences of the agent's interaction. Out of this web of local relations, macrolevel phenomena emerge spontaneously. In their strive for mutual accommodation and self-consistency, groups of intelligent agents '…somehow manage to transcend themselves, acquiring collective properties such as life, thought, and purpose that they might never have possessed individually' [15].

Furthermore, complex self-organising systems are adaptive. Not only do the individual units that comprise the system adapt by constantly recomputing the pattern among the event's dimensions (as was explained in the previous section) but the system as a whole actively also tries to respond to events, which it is presented with. As Melanie Mitchell suggests complex systems '…automatically improve their performance (according to some measure) over time in response to what has been encountered previously' [16]. Similarly to the individual units, the adaptation process at the system level is based on previous knowledge. The knowledge that the system possesses is related to the past performance of its units in dealing with the environment. Endorsing the same type of historical reasoning of their unit, complex systems adapt through differential survival and reproduction of the more successful individuals. The fittest agents are the ones, which have constructed the most efficient strategies to cope with the external needs, spread within the population of units as they have a greater opportunity to reproduce.

The genetic mechanisms utilised by the system to create the new units from the previous generation are mutation and crossover. Through a process of mutation, the genetic characteristics of the fittest agents are inherited by the next generation of units

---

[8] For the study of complexity in biological systems see John Holland and Kauffman. For application of complex adaptive system approach in economics see Albin and Ducan, Brian, and Holland and Miller [17].

[9] For recent works on agent-based modelling in political science see Axelrod , Cederman and Epstein and Axtell [18].

with a certain degree of variation. This allows some opportunity for new units to be created, which may be more effective than their parents in dealing with the external world. Crossing over means the reproduction of new agents by taking some genetic material from one fit parent and some from the other. Through mutation and cross-over of the fittest units, new generations of units are created that display patterns of behaviour that are more likely to resemble those of the more successful agent of the previous generation rather than those of the unsuccessful ones [19]. These main assumptions of complex adaptive systems, self-organisation and evolutionary adaptation, will be utilised as the theoretical foundation for the construction of the multi-agent system.

4. 2 The Evolutionary Adaptive Multi-agent Intelligent System

In the multi-agent system suggested in this paper the single units represent a finite number of Intelligent Adaptive Information System Databases which attempt to make sense of international instabilities using different problem domain models. Each information system is connected to a different theory-driven database supporting a different problem-solving environment. Once a new international problem, for instance, a developing conflict in Asia, is added to the different databases, these single systems would code the problem in different ways and suggest different courses of action. Assuming that no single Intelligent Adaptive Information System Database can provide an effective solution for the problem at stake, collective goals and solution strategies need to be generated by integrating all the outputs of the different systems. A co-ordinator agent that makes use of the evolutionary strategies of complex adaptive systems is suggested as a possible way to produce collective goals and solution strategies out of the goals and solution strategies suggested by the single units. The co-ordinator agent's knowledge comprises of a rule-based component and a case memory, which allows the evaluation of the performance of each system and the creation of a collective solution on the basis of this performance assessment. The case memory will be constantly updated as the multi-agent system is regularly used. The inclusion of new information into the case memory of the co-ordinator agent will provide the multi-agent system with a certain degree of adaptation.

During the process of performance evaluation, the task of the case memory is to store the history of the performance of each Intelligent Adaptive Information System Database, while its rule-based knowledge component computes individual and total fitnesses for each system following pre-defined rules. Suppose that the suggestions produced by a specific Intelligent Adaptive Information System Database were utilised by the management agent in n number of international conflicts, the co-ordinator agent will allocate a fixed value to the performances of this adaptive system. If the management agent following the individual system's recommendation was successful in pursing the suggested objective, the multi-agent system will allocate the value of 2 to the performance of that individual adaptive system. If the goal suggested by the system was partially accomplished by the management actor, the multi-agent system will allocate the value 1. If the management agent was partially defeated in the sense that not only was it unable to achieve the suggested goal but it could also not stop the sides involved in the conflict from accomplishing part of their objectives, the multi-agent system will assign –1 to the system's performance. And finally, if the management agent was defeated in pursuing the suggested strategy and goal, the

multi-agent system will allocate –2 to the individual adaptive system (see Table 1). Given the values of the system's individual performance, a total fitness value, *Total* F_I for each Intelligent Adaptive Information System Database, *i*, is calculated as:

$$Total\ F_i = \sum_{n=1} \frac{f_i I_i}{n} \tag{1}$$

where *n* represents the number of international conflicts in which the management agent followed the system's recommendations, f_i is the system's individual performance in each conflict, and I_i is the importance of the conflict for the international community's interests. International interests are an important factor in calculating the system's fitness, since being successful in solving a situation threatening vital interests of internationals community is more important than being successful in issues not directly related to main international concerns.

Table 1 System Performance Values

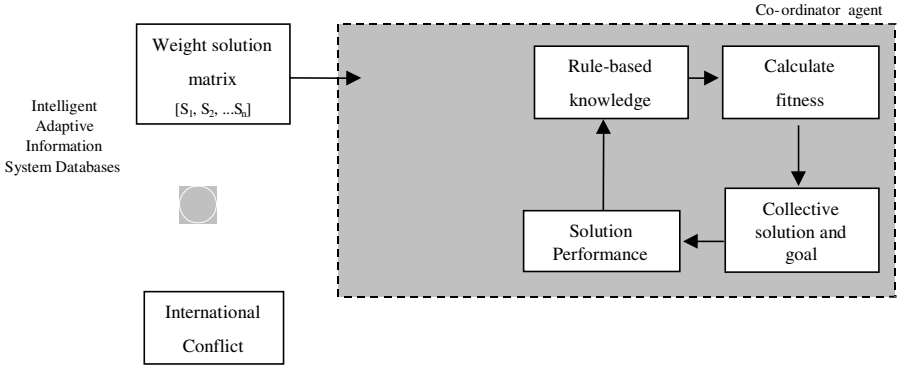
Events	System Performance Values
Management Agent Success	2
Management Agent Partial Success	1
Management Agent Partial Defeat	-1
Management Agent Defeat	-2

The co-ordinator's rule-based knowledge is also used to *generate* the multi-agent system's collective solution. This is done in three phases: first, only the goals and solution strategies of those Intelligent Adaptive Information System Databases whose fitness value is above the average are chosen for creating the collective goal and solution; second, a cross over mechanism is utilised in order to generate the multi-agent system's goals and solutions from the fittest units; finally, the Intelligent Adaptive Information System Databases, which have been selected, participate in constructing the collective goal and solution strategy in proportion to their fitness. Since the solution matrixes of connection weights of the fittest individual units embed systems' goals and solutions for the problem at stake, they are utilised as *genetic material* for the cross over. The Evolutionary Adaptive Multi-Agent Intelligent System's goals and solutions are given by a new solution matrix of weights created by crossing over the weight matrixes of the fittest Intelligent Adaptive Information Systems Databases. As it was said before, the size of the weight matrix taken from each unit in order to create the new collective weight matrix through crossing over would be proportional to the individual system fitness.

The final element of the multi-agent system that needs to be explained refers to its adaptation ability. At the multi-agent system level, the adaptation occurs when the new data related to the performance of each individual system in dealing with a new conflict event is added to case memory of the co-ordinator agent. The new performance achieved by the management agent following the collective recommendation of the multi-agent system, is stored in the case memory and redistributed among the Intelligent Adaptive Information System Databases by the rule-based knowledge of the co-ordinator. The fitness value of the new individual performance for the j^{th} system , f_{ij}, is calculated as follows:

$$f_{ij} = \frac{C_i}{S_{ij}} \tag{2}$$

where C_i represents the management agent's performance in the i^{th} conflict (on the basis of the values of Table 1) and S_{ij} is the size of the weight solution matrix taken from j^{th} unit in order to create the collective weight matrix. This new value is added to the total fitness function, *Total F*, of the j^{th} system (Fig. 4).



References

1. Fischbach, G.: Mind and Brain. Scientific American 267 (1992) 24-33
2. Boyd, R.: The Current Status of Scientific Realism. In: Leplin, J. (ed.): Scientific Realism. University of California Press, Berkeley, CA (1984) 41-82
3. Bechtel, W., Abrahamsen, A.: Connectionism and the Mind. An Introduction to Parallel Processing in Networks. Basil Blackwell, Oxford (1991)
4. Churchland, P.: A Neurocomputational Perspective. The Nature of Mind and the Structure of Science. MIT Press, Cambridge, MA (1989)
5. Davies, M.: Two Notions of Implicit Rules. Philosophical Perspective *9* (1995)
6. Sejnowski, T.J., Kienker, P.K., Hinton, G.E.: Learning Symmetry Group with Hidden Units: Beyond the Percepton. Physica *D* 22D (1986)
7. Rumelhart, D.E., McClelland, J.L., the PDP Research Group: Parallel Distributed Processing. Explorations in the Microstructure of Cognition. MIT Press, Cambridge, MA (1986)
8. Alker, JR. H.R., Bennett, J., Mefford, D.: Generalized Precedent Logics for Resolving Insecurity Dilemmas. International Interactions 7(1982) 165-206; Akihiko, T.: China, China Watching and China Watcher. In: Donald, A. S.,Chan, S. (eds.): Foreign Policy Decision Making: Perception, Cognition, and Artificial Intelligence. Praeger, New York (1984)
9. Beach, L.R.: The Psychology of Decision Making. People in Organizations. Sage Publications, London (1997)
10. Cyert, R.M., March, J.G.: A Behavioral Theory of the Firm. Prentice-Hall, Englewood Cliffs, NJ (1963)
11. March, J., Olsen, J.: The New Institutionalism: Organisational Factors in Political Life. American Political Science Review 78 (1984) 734-49
12. Clark, A.: Moving Mind: Situating Content in the Services of Real-Time Success. Philosophical Perspectives 9 (1995)
13. Most, B., Starr, H.: Case Selection, Conceptualizations and Basic Logic in the Study of War. American Journal of Political Science 26 (1982)
14. Sherman, F.L.: Sherfacts: A Cross-Paradigm, Hierarchical and Contextually Sensitive Conflict Management Data Set. International Interactions 20 (1994) 79-100
15. Waldrop, M.: Complexity. The Emerging Science at the Edge of Order and Chaos. Penguin, London (1992)
16. Mitchell, M.: Imitation of Life. New Scientist (1993) 137
17. Holland, J.: Adaptation in Natural and Artificial System: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, Cambridge, Mass. (1992); Kauffman, S.: The Origins of Order. Self-Organization and Selection in Evolution. Oxford University Press, Oxford (1993); Albin, P., Ducan, F.: Decentralised, Dispersed Exchange Without an Auctioneer: A Simulation Study. Journal of Economic Behaviour and Organisation 81 (1992) 27-51; Brian, A.: Why Do Things Become More Complex? Scientific American (1993); Holland, J., Miller, J.: Artificial Adaptive Agents in Economic Theory. AEA Papers and Proceedings 81 (1991) 365-70
18. Axelrod, R.: The Complexity of Cooperation. Agent-Based models of Competition and Collaboration Princeton University Press, Princeton, NJ (1997); Cederman, L: Emergent Actors in World Politics. How States & Nations Develop & Dissolve. Princeton University Press, Princeton, NJ (1997); Epstein, J., Axtell, R.: Artificial Societies. Social Science from the Bottom UP. Brookings Institution Press, Washington, DC (1996)
19. Holland, J.: Genetic Algorithms. Scientific American 267 (1992)

Intelligent Interface Agents Behavior Modeling

Jorge J. Gómez-Sanz[1], Juan Pavón[1], and Francisco Garijo[2]

[1] Departamento de Sistemas Informáticos y Programación, Universidad Complutense, Ciudad Universitaria s/n, 28040 Madrid, Spain
{jjgomez, jpavon}@eucmos.sim.ucm.es
[2] Telefonica I+D, C/ Emilio Vargas 6, 28043 Madrid, Spain
fgarijo@tid.es

Abstract. This paper presents a software architecture for modeling Interface Agents that is based on the BDI (Believes, Desires, Intentions) approach. This architecture is supported with a methodology that guides the designer to build a complex interface agent. This work has been applied to a fully operative implementation in the EURESCOM P815 project framework.

1 Introduction

The term *Interface Agent* refers to a specific kind of agents [25] that assist users in their interaction with a working system. In essence, Interface Agents provide proactive and reactive assistance in order to increase user productivity. This is achieved, for instance, by teaching how to work with a specific tool (e.g. a data base application) or suggesting better ways to do the same task. The relevance of Interface Agents increases as new applications become more complex. At the same time, users (who do not need to be too familiar with computer technology) do not want to spend a lot of time learning how to use the applications. Also, Intelligent Agents may benefit from the work on Interface Agents as they also need to interact with human users.

Usually, the purpose of an Interface Agent is one of the following:

- Help the user during his work. For instance, the user repeats an action three times without obtaining the desired results (this may be known if the user *undo*es the action). The agent detects this and gives assistance in order to complete the action [16].
- Improve productivity. For instance, the user repeats the same sequence of actions several times (meanwhile the agent is watching him). On trying the sequence again, the agent raises its hand and solicits to perform the sequence by itself [15].

Not as usual, but quite original, is the next one:

- Increase the functionality of an application. From a software-engineering point of view, in many cases the changes needed in an application to increase its functionality are greater than the ones required to add an agent. The investment is strongly justified when distributed and collaborative work is needed to produce results. This approach must be distinguished from the *wrapper* technique [8],

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 598—609, 2000.

whose aim is to transform a legacy application into an agent. With Interface Agents the legacy application stays as it is, and the agents provide the new functionality.

These objectives usually suggest a relationship with the Intelligent User Interface area. Intelligent User Interfaces are human-machine interfaces that aim to improve the efficiency, effectiveness, and naturalness of human-machine interaction by representing, reasoning, and acting on models of the user, domain, task, discourse, and media (e.g., graphics, natural language, gesture) [17]. What is new with the agents is the social ability they have. This ability has been shown in Maxims [16]. Maxims is able to consult other agents Maxims-like to decide what to do when an existing user model is not able to answer our questions.

Many works in Interface Agents attempt to apply user-modeling techniques in order to give the agent some learning abilities. The most common techniques are:

- A set of (Situation, Action) pairs. Maes [16] applies memory reasoning techniques as follows. She considers all possible variables that represent the situation of the user. When the user acts, the action is coded and the variables are instantiated using the current situation, so that a new pair (action, situation) can be created. This way, the next time that the same situation appears, the agent will suggest to perform the same action the user did before.
- Techniques based on Similitude Functions. Mitchell [18,19] considers the problem of relevant document selection according to some examples given by the user. The technique used is *vector space model* [24] using similitude functions based on the *Mutual Information* [21] concept.
- Decision Trees. In [18] decisions are taken using a decision tree built following an algorithm similar to *ID3* [24].
- Statistical models [13]. The idea is to predict the future actions of the user through probabilities. This probabilities are generated by a statistical model which uses data about the past user actions.

Although several Interface Agents have been already presented in the literature, we did not find well-documented architectures for implementing this kind of systems. A high percentage of current Interface Agents base their control on simple heuristics and are specific to a domain. So the papers describing these works do not address the software architectural issues. We have only found in COLLAGEN [22, 23] a generic way for modeling Interface Agents, which is based on the *discourse theory* [9] to model the interaction between an entity, the agent, and the user. Cohen [2] presumes that his architecture can also be applied to Interface Agents. Nevertheless he does not seem to have progressed in that direction.

BDI has been extensively used in different applications of AI. Our purpose is to define a generic software architecture for modeling Interface Agents that is based on the BDI approach [10]. Defining the behavior of an interface agent using a BDI model eases the design and maintenance of Interface Agents (see Conclusions). We present the architecture of the cognitive system for implementing the behavior of the Inteface Agent (see Section 3). This architecture is supported with a methodology

that guides the designer to build a complex interface agent (Section 4). This is illustrated with an example, which has been applied in a prototype in EURESCOM P815 [4] (Section 5). Finally, the conclusions discuss the benefits of Interface Agents and open issues for future work.

2 Modeling Interface Agents with BDI

BDI architectures are inspired in a cognitive model of humans [1]. They use a model of the world, a representation of what the world looks like for an agent. Agents receive stimuli through sensors situated in the world. These stimuli modify the model of the world that the agent has (represented by a set of Believes). To guide their actions, the agent has Desires. A Desire is a state that the agent wants to achieve through the Intentions. Last ones are special actions that can be aborted due to changes in the model of the world.

There are other BDI architectures [10] already implemented. Each one has a different conception of how to reflect this model. Nevertheless, all of them agree that there is a need to:

1. Point out which intentions are going to be performed.
2. Specify the goals.
3. Model the world.

The *desires* of an agent, in some cases known as *goals*, wait for the appropriate believes to happen so that they can compromise themselves through intentions [10]. An Intention can be defined as an interruptible process that performs actions over the real world. It usual to include another element in these architectures: the *Plan*. It is a high level organisation of the desires of the agent. A *plan* makes possible to co-ordinate agent's actions together with the current world model.

The application of the BDI model to the development of Interface Agents makes the following conceptual mapping:

- *Believes* as environment events. The agent starts from a model of the world (the application it collaborates with) that consists of the initial screen (the application has just been launched). This model is modified by the events coming from the user, the application or other agents. So, these events can be considered as believes about new states of the world.
- *Desires* as agent's goals. The idea of desire is included in the agent's design. Meanwhile the user is working with the application, the agent is looking for ways of providing assistance. In order to accomplish this objective, the agent creates some high level goals (e.g., letter redaction assistance, curriculum vitae fulfilment assistance, etc.). These high level goals can be divided into simpler ones to get more realistic tasks (e.g., retrieve letter examples that resembles to what the user is writing, correct spelling mistakes, etc.).
- *Intentions* as the actions of the agent. A change occurred in the world model can make current intentions no longer valid. For example, the action of letter redaction assistance is an intention that can be aborted if the agent window is

closed. The closure is translated as "do not help me to write letters". Therefore, the letter redaction assistance intention must be aborted.

- *Plans* as the co-ordination of agent's actions. The Interface Agent has some ideal states (goals) that it wants to achieve. To reach those states, the agent performs actions according to the current situation. In BDI paradigm, agents use *plans* to cover these issues. The problem is the same as to co-ordinate world knowledge with desires and intentions.

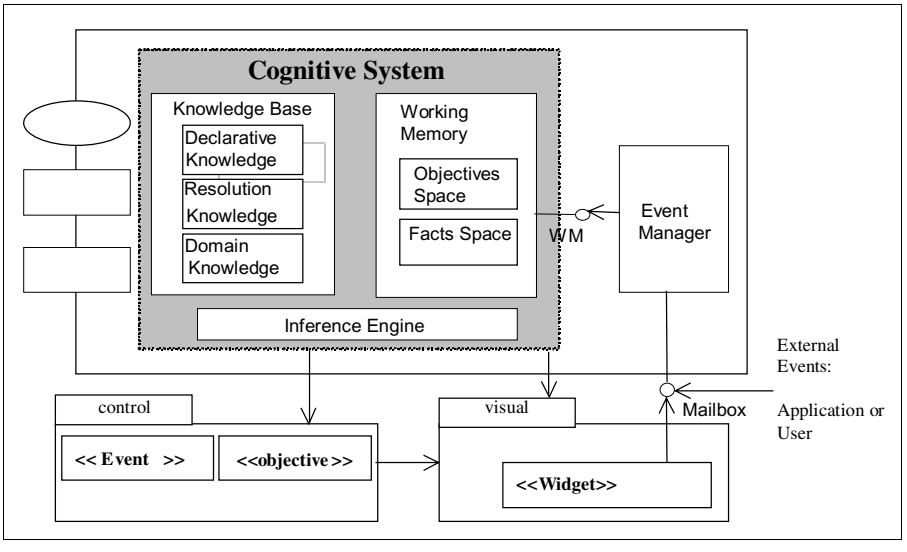


Fig. 1. Interface Agent Control Architecture. Events are asserted into the Inference Engine, and processed using the Knowledge Base.

3 Architecture of the Agent

The basic architecture of the agent is inspired in [7] and was first used in EURESCOM P712 [3]. In the original model, the control of the agent is considered to be symbolic (it was based on the use of a rule based system). Nevertheless, the actions were implemented in a non-symbolic way (imperative language). We have added to the original model the concept of *space of objectives* (described below) and improved different aspects of agent behaviour modelling as well as the implementation process: behaviour rules reengineering, structures to handle recursive intentions, intention reengineering and application of object oriented design patterns [6].

Agent's behavior is implemented with a cognitive system that is fed by events from the environment (i.e., user actions, events from the application, or messages from other agents), as shown in Figure 1. The cognitive system consists of an inference engine (JESS [11]), a knowledge base and a working memory. Incoming events can be asserted as facts in the working memory. Then, the application of behavior rules, which are defined in the Knowledge Base, enable the intention performance.

The knowledge base defines the behavior of the agent. It is structured in:

- *Declarative Knowledge*, which defines the inference rules to handle refinement, resolution, validation and failure. It defines the relationship between the events and the goals of the agent. These goals are declarative predicates that point at some computational objects called *objectives* (described below). Using the properties of the inference engine, the objects are asserted as goals.

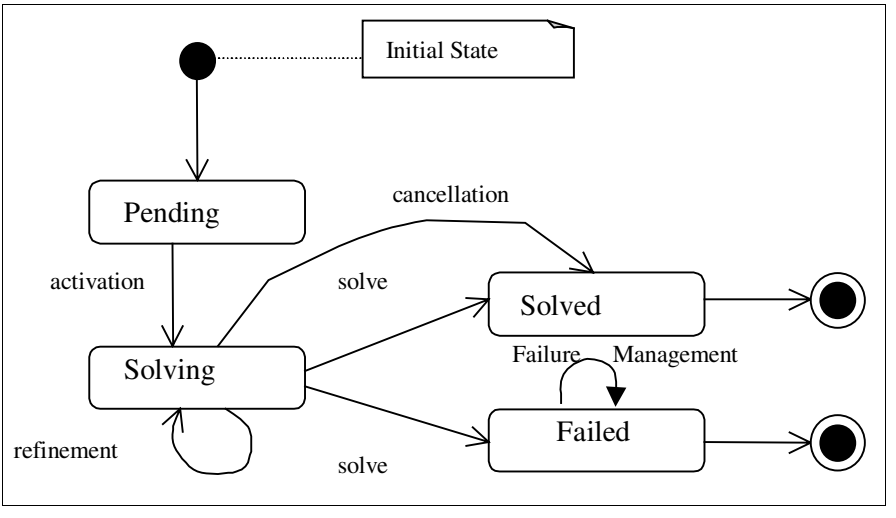


Fig. 2. These are the states in a *objective*'s lifecycle. The notation used is UML. From an initial state, the *objective* receives transition orders so that a complete intention is performed. Note that a failure state means that the intention has been aborted.

- *Resolution Knowledge*, which consists of a set of computational objects, the *objectives*, that at this level represents the intentions of the agent. They are Java classes including the resolution actions (*refinement*, *activation*, *solve*, and *failure management*) that will be invoked along the *objective's* lifecycle (see Figure 2). Note that there is one action associated with each state transition.
- *Domain Knowledge*, which consists of all classes that represent domain entities that may be handled by the agent (for instance, window descriptions, communication facilities, etc.)

The Working Memory represents the status of the world during agent's execution. This information is structured into:

- An *Space of objectives*. It represents the set of goals, and their relationship, that the agent has in an instant of time. Basically, it is and AND/OR graph [24] whose size and structure will change depending on the user actions, application status or other agent's actions. An *space of objectives* is composed of a *root objective* and several *space of objectivess* that represent subgoals to achieve in order to solve the *root objective*. Each *Objective Space* is responsible to decide whether its *root objective* can be solved or not.

- A *Space of facts*, contains the evidence and intentions of the user in a specific moment (for instance, events that have occurred).

The improvement that has been done from [7] and [3] allows modifying or enlarging the agent's behaviour with a minimum impact in the existing code. This has allowed a progressive grow up of our first Interface Agent prototype, what it is not usual as many developers know. With such an agent, it has been possible to apply a spiral project engineering live cycle, which is mostly remarkable.

The components of this cognitive model map to BDI concepts presented in Section 2 as follows:

- Desires as *objectives* predicates. The computational objects *objectives* are asserted in the inference engine as predicates. Inside the inference engine, there is a relationship between them and the events. They can be considered as goals that wait for certain believes to happen.
- Intention as an *objective*. The behavior of an BDI intention is represented in the live cycle of an *objective*. The action, represented by a sequence of an *objective*'s method calls, can be aborted, changing the state of the involved *objective* to *failed*. Or it can be finally successful, changing the state to *solved*.
- Believe as *event*. Events are the information that allows to model the external world. Their function as believes has been explained in Section 2.
- *Plan* as the *space of objectives*. A *plan* indicates what to do next so that certain goal can be accomplished. In this case, what was needed was an object responsible to manage relationship between *goals* and to say what to do when a subgoal was *failed* or *solved*. Using a fixed *plan* was not enough (*objectives* had to be watched and the environment's dynamic nature considered). So it was created the *space of objectives*.

4 Methodology to Develop a BDI Based Interface Agent

This is a very basic description of a methodology to build Interface Agents based on the proposed architecture (Section 3). It intends to be a practical guideline to those attempting to apply this architecture to a real Interface Agent development.

The definition of a BDI model for an Interface Agent and its implementation is done in three steps:

1. Identify events, intentions and resources. This is the *Domain Knowledge*.
2. Organize actions inside *objectives* and definition of the agent's *space of objectives*. This is the *Resolution Knowledge*.
3. Determine which believes (events) allow to execute each part of the plan (*space of objectives)*. This is the *Declarative Knowledge*.

In the first step, it must be clear which actions the agent will commit, which resources has available and how they can be used. With this architecture, shared resources problems arise naturally, since this is an inherently concurrent system (every rule that handles intentions can be executed concurrently). So, first of all, the

shared elements of the domain must be thread safe. With regards to user behaviour and the application, they both must be modelled, at least briefly, in order to know in which cases the agent will act. This is achieved by identifying the events that allow the evolution from the agent's initial model of the world to the desired situations (represented by another model of the world). It is recommended to create scenarios where events, resources and intentions appear. These scenarios should show whether concurrent processes might need synchronisation mechanisms. From these scenarios, the domain objects, like resources, can be immediately designed.

In the second step, the intentions obtained in step 1 must be refined and organised to:

1. Fit into an *objective*. An intention can be complex enough to be decomposed in simpler actions. An *objective* can contain only one intention.
2. Fit into the lifecycle (Figure 2) of an *objective*. Intentions must be divided into stages to fit into the *objective*'s lifecycle. For example, if an intention needs some resources before its completion, they have to be obtained during an *activation* transition; and the completion itself have to be done during a *solve* transition.

Objectives can be refined, inside an *space of objectives*, to achieve other concurrent fine-grained *objectives* (*divide and conquer* strategy [24]). The programmer sets the dependence between the *objectives* and the created *subobjectives*. During EURESCOM P815 [4] two dependence types were needed:

- AND dependence: an *objective* can be solved if all of its *subobjectives* are solved
- OR dependence: an *objective* can be solved if any of its *subobjectives* is solved

These *objectives*, organised following an *is-subtask-of* relationship, constitute something resembling a *plan* (see Section 2).

In the third step, the *objectives* are inserted in the control of the agent as rules. For each rule, the left-hand side describes the conditions (events, *objectives's status*, *facts*, …) under which the rule is fired. The right hand side describes the intention as call to an *objective's* lifecycle transition.

5 Practical Example

This architecture and methodology for Interface Agents has been applied in the project Eurescom P815 [4] for the definition of agents that increase the functionality of a legacy workflow management system, an Intelligent Network Service Creation Environment by Telefonica I+D.

Project EURESCOM P815 studies how agents can improve a business workflow process. One of the experiments in the project has been to create Interface Agents that collaborate with a legacy workflow management system, a Service Creation Environment (SCE) for Intelligent Networks. In the SCE, the business workflow process is determined by the activities that the participants of a service creation process can do. The participants in the workflow are the Project Manager, the Project Developers, and the Project Reviewers. For the first two, an Interface Agent has been defined: Project Manager Assistant (PMA) and Project Developer Assistant

(PDA). These agents can collaborate to increase productivity of these types of users by adding new functionality to the SCE, and by improving communication between Project Managers and Project Developers. The SCE offers the basic functionality and data to the Interface Agents so that they can accomplish their goals. The SCE consists of several CORBA objects to which the agents have access. The most relevant of them are the following:

- *Project Management*. It provides basic functionality for a Project Manager.
- *Project Development.* It provides basic functionality for a Project Developer.
- *Workflow Management System.* It manages the workflow of the project.
- *Project Management Assistance and Project Development Assistance*. They contain the infrastructure of the agents.
- *Development tools.* They are shared tools, like emacs or Framemaker
- *Information Repository.* It contains the project, users, tools and project lifecycle data.

The example selected for this paper is a monitarization problem. There is a task to be done in a project. Its assigned developer is not going to have enough time to finish it. Several reports about the status of the task have been sent to the Project Manager; so he knows the situation of the involved task. The aim is to provide the PDA with functionality to foreseen when a task is not supposed to be finished on time. Applying the steps shown in Section 4:

Step 1: The resources needed are the information repository, to obtain an updated task status, and the reports sent to the Project Manager. The reports are part of the extra functionality provided by the PDA and PMA. So, the PDA has access to both resources. The event to consider is the proximity of the deadline. To obtain the event, as it is not originally considered in the application, there must exist an object (inside the agent) in charge of checking the deadline proximity. Let us call this object *scheduler*. As soon as 80% of assigned time for a task has been consumed, the *scheduler* generates an event for the agent. The intention is to obtain the data (task's reports and task's status) and:

1. Warn the user if the task is not going to be finished on time.
2. Warn the user the deadline is near, but that his work is progressing correctly.

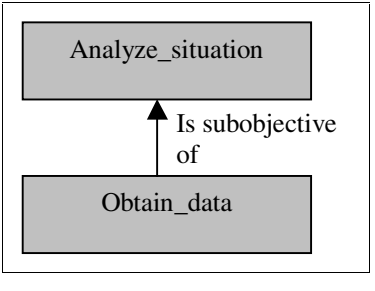


Fig. 3. Objectives that analyze the task's status. They have an is-subobjective-of relationship. The main *objective* is *Analyze_situation*. In order to be solved, it needs that previously *Obtain_data* is solved.

Step 2: the intention is refined into two intentions: to obtain the data and to analyse it. Both could be handled in one *objective* (with an *activation* and a *solve* transition). Nevertheless, the action of obtaining data could fail: failure management should be included for *activation* transition too. In order to fit into the *objective's* lifecycle, two *objectives* will be created (Figure 3) and each one will have its own failure management. The relationship between them can be either AND or OR (they are equivalent relationships when there is only one *subobjective*, see Section 4). It must be clear that no analysis can be done before obtaining the data.

Table 1. Rules to track the proximity of a task's deadline. They are part of the Declarative Knowledge (Section 3).

Rule 1	**If** There is an event indicating proximity of a deadline for a task **and** there exists ***Analyze_situation*** in PENDING state **then** tell the ***objectives space*** to refine ***Analyze_situation***
Rule 2	**If** there exists ***Obtain_data*** in PENDING state **then** activate ***Obtain_data*** **and** solve ***Obtain_data***
Rule 3	**If** there exists data (reports and status) about a task **and** there exists ***Analyze_situation*** in SOLVING state **then** solve ***Analyze_situation***

In *Obtain_data*, during *activation* transition, CORBA connections will be established with the *Information Repository.* During *solve* transition, the whole data will be retrieved and asserted into the inference engine.

During *refinement* transition, *Analyze_situation* is refined into *Obtain_data*. During *solve* transition, the data retrieved will be analysed and the proper actions taken (notify the user).

Step 3: the events are associated to objectives indicating what stage of the intention can be executed. Table 1 shows what kind of entries can be used to handle the task deadline monitorization problem.

Each new case study implies the definition of new objectives, events and rules, that are added gradually to the cognitive system, following the 3 steps of the methodology.

6 Conclusions and Future Work

The main benefit of BDI models that we have found is reduction of development and maintenance cost. Regarding development cost, there are two conclusions:

- There is an implicit architecture in the model (Section 2). This eases the development process as there is a framework to put the agent's specific code in. Many of the Interface Agents in the literature do not specify whether there is or not a well-founded architecture behind the code (see Introduction). With light agents, this can be acceptable but with large ones, and from a software-engineering point of view, it is not at all recommended.
- There is an easy mapping of domain elements to known BDI structures (Section 3). It is most desirable that developing Interface Agents become a matter of typing specific code for the desired domain.

And regarding maintenance cost, if more functionality has to be added to an Interface Agent, an underlying BDI architecture, in this case, focuses changes in specific components. Usually, adding functionality means adding more Desires, Believes and Intentions to the system (as in Section 5). So, there no need to change the structure of such an Interface Agent.

Summarizing, we found BDI based agents:

- *Robust*. The agent acts just when and how it is supposed to. During the design stage, designer must define Believes that the agent will recognize and how to react when they appear. This includes the failure situations to which a plan can lead [12].
- *Simple*. BDI designs are simple. They follow the operative sequence: "obtaining believes", "processing believes", and "acting over the world". This way of reasoning, as it is very close to the one of human beings [1], leads to clear and intuitive designs.
- *Extensible*. The agent can increase the number of tasks that can accomplish in a convenient way. It is just a matter of adding new Desires, Intentions, and Believes into the architecture.
- *Reusable*. The BDI architecture designed for an agent can be reused in different domains. What changes is the intentions, the believes and the rules

There are previous works on the use of a BDI model for agent design, most of them from the *Australian Artificial Intelligence Institute*. However, no one addresses the Interface Agent development (except cases shown in Introduction). This shortage seems logical, since most of the research on Interface Agents is focusing in how to obtain intelligence (heuristics). In this article, it has been shown that a BDI approach can report several benefits to the Interface Agents development. It was also presented a way to model the behavior of such agents following the BDI model, as a result of our expertise in EURESCOM P815 [4].

Future work is centred in the improvement of the social ability and intelligence of the developed Interface Agents. They are remarkable qualities in the agency concept that need reviewing in the current design. One approach to do this is by adding *user*

modelling and *learn by example* techniques, following Maes [16] and Lieberman [13, 14, 15] work. With regard to the social ability of the agent, the next step is to adapt to the FIPA [5] communication languages and protocols in order to be FIPA compliant.

Acknowledgements and Disclaimer. The authors wish to thank Telefónica I+D and EURESCOM for their support.

References

1. Bratman, M.E.: *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA. (1987)
2. Cohen, P.R.,Cheyer, A.,Wang, M., and Baeg, .S. C.: *An open agent architecture*. AAAI Spring Symposium: Pages 1-8. (1994)
3. EURESCOM P712: *Intelligent and mobile agents and their applicability to service and network management*. Published in *http://www.eurescom.de/Public/Projects/p700-series/P712/P712.HTM*
4. EURESCOM P815: *Communications Process Integration Using Software Agents*. Published in *http://www.eurescom.de/Public/Projects/p800-series/P815/p815.htm*
5. FIPA: *Foundation for Intelligent Physical Agents*. Published in htt://www.fipa.org
6. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patters: Elements of Reusable Object-Oriented Software*. Addison-Wesley. (1994)
7. Garijo *et al.*: *Development of a Multi-Agent System for cooperative work with network negotiation capabilities.*Aparecido en *Intelligent Agents for Telecommunication Applications*. LNAI 1437. Ed. Springer Verlag. Pages 204-219. (1998)
8. Genesereth, M. R., and Ketchpel, S. P.: *Software agents*. Communications of the ACM, 37(7): Pages 48-53. (1994)
9. Grosz, B. J., and Sidner, C. L.: *Attention, intentions,and the structure of discourse*. Computational Linguistics 12(3): Pages 175-204. (1986)
10. Georgeff, M., Pell, B., Pollack, P. E., Tambe, M. y Wooldridge M.: *The Belief-Desire-Intention Model of Agency*. Intelligent Agents. Springer Publishers. (1998)
11. JESS: Java Expert System Shell System. Published in *http://herzberg.ca.sandia.gov/jess*
12. Kinny, D., Georgeff, M., Rao, S.: *A methodology and modelling technique for systems of BDI agents*. Tech. Rep. 58, Australian Artificial Intelligence Institute, Melbourne, Australia, January 1996.
13. Lieberman, H.: *Letizia: An Agent That Assists Web Browsing*. Proceedings of the 1995 International Joint Conference on Artificial Intelligence, Montreal, Canada, (1995)

14. Lieberman, H.: *Integrating User Interface Agents with Conventional Applications*. International Conference on Intelligent User Interfaces, San Francisco, January 1998.
15. Lieberman H., Maulsby D.: *Instructible agents: Software that just keeps getting better.* Systems Journal, Vol. 35, N. 3,4. (1996)
16. Maes, P.: *Agents that Reduces Work and Information Overload.* Aparecido en Readings in Intelligent User Interfaces. Morgan Kauffman Publishers, Inc. (1998)
17. Maybury, M.: *Intelligent user interfaces: an introduction*. Proceedings of the 1999 international conference on Intelligent user interfaces , Pages 3 – 4. (1999)
18. Mitchell T., Caruana R., Freitag Dayne (1994): *Experience with a Learninq Personal Assistant*. Communications of the ACM, July 1994, Vol 37, No. 7: Pages 81-91. (1994)
19. Mitchell, T.: *WebWatcher: Machine Learning and Hypertext*. Report from School of Computer Science, Carnegie Mellon University, (1995)
20. Mladenic, D.: *Personal WebWatcher: Implementation and Design*. Technical Report IJS-DP-7472. Carnegie Mellon University. (1996)
21. Quinlan, J.R.: *C4.5, Programs for Machine Learning*. Morgan Kauffman. (1993)
22. Rich, C., Sidner, C.: *Adding a Collaborative Agent to Graphical User Interfaces*. ACM. Symposium on User Interface Software Technology. (1996)
23. Rich, C., Sidner, C.*: COLLAGEN, when Agents Collaborate with People*. First International Conference on Autonomous Agents, Marina del Rey, C.A. (1997*)*
24. Rich, E., Knight, K.:*Artificial Intelligence*. McGraw-Hill. New York. 2nd. Edition (Spanish edition), (1991)
25. Sycara, K: *The Many Faces of Agents*. AAAI Magazine. Summer 1998.

Memory Based Reasoning and Agents Adaptive Behavior

Ana S. Aguera[1], Alejandro Guerra[2], and Manuel Martínez[2]

[1]Instituto de Investigaciones Eléctricas
Av. Reforma 113
Temixco, Morelos.
México
aaguera@iie.org.mx
[2]Departamento de Inteligencia Artificial
Universidad Veracruzana
Sebastián Camacho # 5
Xalapa, Veracruz
México
Tel (28) 17 29 57 Fax (28) 17 28 55
aguerra@mia.uv.mx, mmartine@mia.uv.mx

Abstract. The main purpose of this work is to explore the application of Memory Based Reasoning (MBR) to adaptive behavior in agents. We discuss the design of an interface agent for e-mail management assistance as a prototype for the experimental assesment of an MBR learning algorithm performance. Results are discussed and a brief summary of conclusions follows, as well as a sketch of future work to be done.

1 Introduction

In this paper we explore the possibility of applying Memory Based Reasoning (MBR) to adaptive behavior in interface agents. We start by giving a definition of what is to be understood as an interface agent, followed by a discussion about the importance of learning in order for the agent to adapt to the interface user behavior. Next a detailed description of the MBR learning algorithm is given, followed by discussion of Hermes-II, an interface agent prototype for e-mail management assistance. The final sections are devoted to the presentation of experimental results and a summary of the main conclusions obtained.

Our work is related to Maes and Kozierok agent which learns from three different sources: User observation, user feedback and explicit training from the user [Maes and Kozierok 93]. The agent proposed by Gary Boone [Boone 98] learns concepts extracted from e-mail useful features, and later applies these concepts to learn the user actions.

The method here described does not generate rules or concepts, uses only one source for learning, and as shall be shown, has a high degree of predictability and a high learning rate [Aguera 99].

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 610—620, 2000.

2 Interface Agents

The agents we consider are part of an embedded system [Kaelbling 93]. An embedded system is a pair agent-world. The *world* or *environment* may be described by a 3-tuple <E,A,T> where E is the set of possible *world states*, A is the set of *actions* the agent can take on the world, and T is the world *transition function* T:EXA→E, which maps each pair (state,action) into a new state. T may be deterministic or stochastic, defined by a family of conditional probability functions. In general, an agent may be defined as a 3-tuple <P,R,B,> in which P is the set of *perceptions* of the agent (or the set of the world states that the agent distinguishes), R is the *reinforcement* or *reward* function R:E→R which assigns to each state a real number (R denotes the set of real numbers), and B is the *behavior function* B:(P,R)→A which maps each pair perception-reward into an action (in many applications E and P are identical). R is related to the agent goals. In fact this formalism is related to the so called *PAGE* description (*P*erception, *A*ction, *G*oals, *E*nvironment).

The system then evolves as follows: the agent selects an action a, given that the world is in state e, then the world moves to a new state e'=T(e,a), which has associated a reward r, the agent percieves e' as p' and selects a new action a'=B(p',r), the cycle repeats and the agent is said to behave uninterruptedly.

It is said that the agent exhibit learning behavior if it has a built-in algorithm that modifies function B in order for the agent to acquire (or approximate) maximum reward, usually given by the expected total reward [Kaelbling 93]. This is considered to be the agent´s goal.

An interface agent [Maes 94], [Nwana 96], [Mitchell et al. 94], [Laurel 90] is a software component complementary to a user interface, in which agent and user engage in a colaborative process which includes communication, event monitoring and task ejecution. The agent's world is the interface which in turn is the user-agent interaction medium. The agent acts independently of the interface in benefit of the user´s goals and without his explicit intervention [Lieberman 97]. The metaphore for this kind of interaction is that of a personal assistant [Maes 94].

A personal assistant must have a great deal of autonomy so that it can act independently in the ejecution of specific tasks, without constant external intervention or supervision. It must be robust so that it is less vulnerable to failure, and it should have some knowledge about the user's habits and preferences. Since in general the user's behavior changes in time, so that it is not always possible to predict his future actions by means of fixed rules, the agent must exhibit some kind of adaptive functionality in order to be useful.

In this paper we present a learning algorithm based on MBR that makes an agent adapt to user's needs, illustrating its use through the prototype Hermes-II, an e-mail management assistant. The following section describes the basic MBR algorithm used throughout.

3 Memory Based Reasoning

Different MBR algorithms can be found in the literature [Stanfill and Waltz 86], [Kasif et al. 97], each one with its own specific features depending on the application domain.

Nevertheless a general inclusive description of MBR algorithms may be given.

An MBR algorithm acts on the following components:

- *Mem:* a memory containing N examples,
- X : a new example described by the vector $\{x_1,x_2,...,x_m\}$ where $\{x_1,x_2,...,x_{(m-1)}\}$ are the prediction attributes and x_m is the class attribute,
- Y : a memory example $\{y_1,y_2,...,y_m\}$, with the same structure as X,
- V_k : a vector containing the k examples in memory closest to X.

The algorithm works as follows:

1. For each example Y in *Mem* compute the distance $d(X,Y)$ (the distance function employed in our work is defined below).
2. Search for the set V_k (using a specific criterion for selecting the k-closest neighbors to X in *Mem*).
3. For each element in V_k obtain the values of the class attributes $y_{m's}$.
4. Select the class value y_m that best predicts x_m .

Once a prediction has been made, a confidence measure is computed. We used a confidence measure proposed by [Maes 93]:

$$\left(1 - \frac{\frac{d_{predic}}{n_{predic}}}{\frac{d_{other}}{n_{other}}}\right) * \frac{K_{total}}{K}$$

where (vector $\{x_1,x_2,...,x_{(m-1)}\}$ is called the new situation):

- K is the number of cases considered to make the prediction of x_m
- d_{predic} is the distance of the attribute vector $\{y_1,y_2,...,y_{(m-1)}\}$ in V_k closest to $\{x_1,x_2,...,x_{(m-1)}\}$ with the same class attribute y_m as the predicted one.
- n_{predic} is the number of cases closest to the new situation within a treshold value with the same class attribute as the predicted one.
- d_{other} is the distance to the closest example with a different class attribute.
- n_{other} is the number of cases closest to the new situation within a threshold value with class attribute not equal to the predicted one.
- K_{total} i s the sum $n_{predic}+n_{other}$.

If the result is negative, the confidence is 0. If there is no n_{other} value then we set the confidence equal to 1.

We used as distance function the weighted value difference metric (**wvdm**) defined as follows.

Let X be a new attribute vector, Y an example in Mem and $G=\{g_1,g_2,...,g_n\}$ the set of possible values of the class attribute (x_m). The distance between an example Y and the new vector X is

$$d(X,Y) = \sum_{i=1}^{m-1} vdm(x_i, y_i)$$

The value difference between two values x and y of a given attribute is computed by

$$vdm(x,y) = \sum_{g \in G} \left| \frac{f_g(x)}{f_m(x)} - \frac{f_g(y)}{f_m(y)} \right|^2$$

Where

- $f_g(x)$ is the frequency of x within all cases having the g-th class value.
- $f_m(x)$ is the frequency of x in all memory examples.

Since this metric assumes attribute independence, then the ratio $f_g(x)/f_m(x)$ is an estimate of the conditional probability $p(g/x)$ that value g occurs given that the attribute considered takes value x. It is important to weight the relevance of attributes, which can be attained using wieghts:

$$w(x) = \sqrt{\sum_{g \in G} \left| \frac{f_g(x)}{f_m(x)} \right|^2}$$

Thus, the **wdvm** is given by

$$d(X,Y) = \sum_{i=1}^{m-1} vdm(x_i, y_i) w(x_i)$$

4 What Is Hermes-II?

Hermes-II is an interface agent in experimental development, its task is to assist a user in e-mail management incorporating MBR to learn the user's behavior and eventually automate repetitive actions of the user. We present the PAGE (Perception,Action,Goals,Environment) description of the agent and discuss what the agent has been able to achieve.

By e-mail management we mean the presentation and administration of e- mail messages. In a first version (Hermes-I, [Aguera and Guerra 98]) the agent's task was simply to retrieve messages and show them to the user. Hermes-II incorporates the ability to learn from the user's behavior how to manage new messages, suggesting or deciding what to do with them (store, delete, answer, etc.) according to user's past preferences. Hermes-II stores its experience in a memory which is used as the basis of the MBR algorithm. Wenever a new mail message arrives Hermes-II tries to select an appropiate action (store, delete, etc.) according to past user´s behavior. To achieve this it selects those examples closer to the new one and chooses the most promising action, in terms of the criteria discussed in the MBR algorithm section. Next, the agent computes the predictive confidence and, given a threshold, decides either to take an action, make a suggestion to the user, or make no suggestion and wait until it gathers enough information. Hermes-II interacts with the user through a graphic interface.

4.1 PAGE Description of Hermes-II

- **Perceptions**

Hermes has three sensors to gather information from its environment:

- User's Information (UI*):* this sensor allows the agent to identify the user and locate his mail box, thus

$$UI=(server,login,password)$$

- *New Mail (NM)*: This sensor detects the arrival of new messages, it is a binary sensor:

$$NM = \begin{cases} 0 & \text{if there are no new messages in the mail box} \\ 1 & \text{if there are new messages in the mail box} \end{cases}$$

- *User Action*: This sensor recognizes when the user has taken a particular action over an active message, the parameters delivered by this sensor are:

$$UA = \begin{cases} (message, action) & \text{if a button of the GUI is activated} \\ null & \text{otherwise} \end{cases}$$

Where *message* is a message description and *action* is the action associated with the selected button.

- **Actions**

The set of Hermes-II actions is subdivided in *basic actions* and *high level action*s.

- *Basic actions* are defined as those that the agent can activate on the user e-mail interface: read,delete, answer, forward, store.
- *High level actions* are those defining the behavior of the agent: *reasoning* and *learning*. Reasoning generates the basic actions and learning adapts the results of reasoning about the user's behavior.

- **Goals**

Hermes-II has as a global goal to adapt to the user's behavior in such a way that the user is satisfied, that is that the user rejects the smallest number of suggestions made by the agent. Thus Hermes-II should try to minimize the ratio

$$N_r/N_s$$

computed from the last p suggestions, where N_r is the number of rejections and N_s is the total number of suggestions. To minimize the above ratio is equivalent to maximizing

$$(N_s-N_r)/N_r=1-N_r/N_s$$

which is an estimate of correct prediction probability.

- **Environment**

Hermes-II environment is conformed by the e-mail application, the system which it resides and the user with which it interacts through the GUI. The environmet dynamics is defined by two sources: changes in the environment produced by the system (eg.

message arrivals) and the actions taken by the user. The environment is non-deterministic since it is not possible to predict its next state given the present perception of the agent (for example, it is not possible to know what the user's next action will be, or if there will be a new message at all). Figure 1 shows the interaction of Hermes with its environment.

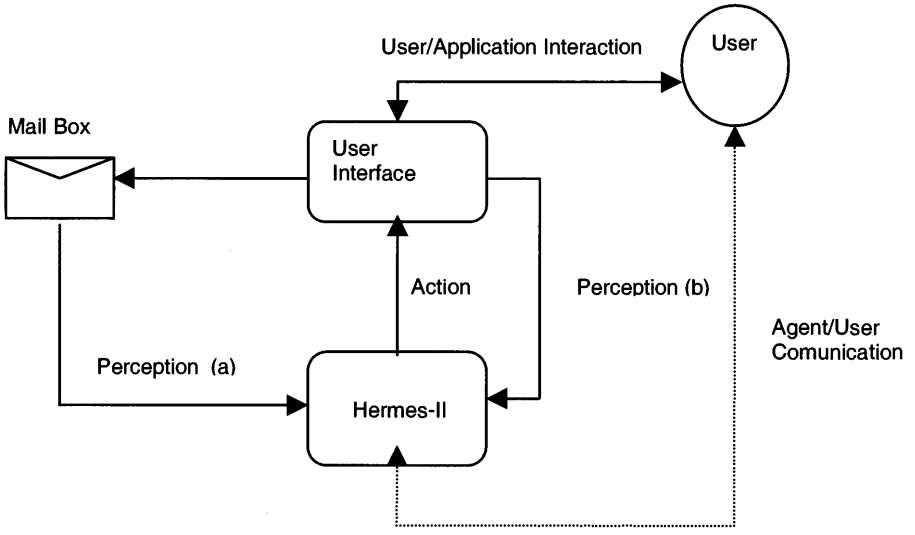


Fig. 1. Hermes-II through its senses can have two types of perceptions: (a) new message arrivals and (b) user actions on the active mesage. Each of these types activates a high level action that eventually triggers a set of basic actions.

4.2 Environment States and Transition Function

Since the environment is nondeterministic it is not possible to make a precise description of the transition function, although it is feasible to assume a probabilistic transition function, such as we did for the simulation study as described later.

4.3 Learning Elements

Hermes-II learning element is the MBR algorithm described earlier. Hermes-II selects actions by *recalling* past situations recorded in its memory. Figure 2 shows the interaction between the agent's perceptions and the learning algorithm.

5 Results and Conclusions

There is a user model underlying the MBR algorithm [Stanfill and Waltz 86]. This model assumes that the attributes are conditionally independent given the user's action; that is the user's behavior can be modelled by a simple bayesian network. Using this

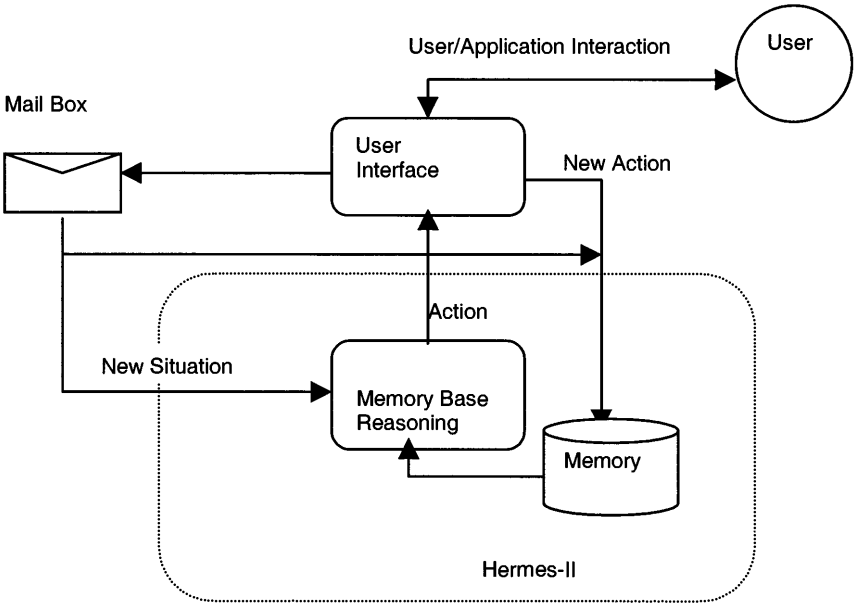


Fig. 2. The arrival of a new message activates the reasoning mechanism to generate a basic action. A new action by the user together with the new situation updates Hermes-II memory.

fact we were able to simulate several users with different behavior patterns by making use of a bayesian network simulator [Aguirre 98]. We also applied Hermes-II to two real users. As a measure of performance we used the *predictive confidence*, as earlier defined, the *percentage of correct predictions* defined by

*100*Number of correct predictions/Total number of predictions*,

and the *accepted suggestions (among the last p) proportion* given by

(1-Nr/Ns)

The two first measures are related directly with the global agent's behavior, while the last one relates the predictive confidence with the agent's ability to make a right guess.

We now present two examples and the results obtained. The first user is a simulated user, the second one is a real user.

5.1 Simulated User

In our simulations we assumed that the user shows regular patterns of behavior only for some attribute values, for example whenever she or he receives an e-mail which is known to be a comercial advertisement (say by looking at the *sender*) then he or she immediately deletes it without even opening it. In other cases he or she receives an e-mail from a colleague, then depending on the subject he or she will either to read it immediately or store it for later reference. We call the examples belonging to this class *structured examples*. There some other cases in which the user does not follow any particular behavior pattern, we call these cases *unstructured examples*. For real users it

is not possible to distinguish a priori these two classes, but anyway the agent learns the stuctured behavior and has a good overall performance.

In this example we simulated a user with the following structured (probabilistic) behavior:

Sender	Day	Time	Subject	Action	Probability
aguerra	*	*	*Hola *	Read	0.9
aguerra	*	*	*Hola *	Reply	0.1
aguerra	*	*	*Saludos *	Read	0.9
aguerra	*	*	*Saludos *	Reply	0.1
aguerra	*	*	*Tesis *	Save	1.0
aguerra	*	*	*Avance *	Save	1.0
aguerra	*	*	*Agentes *	Save	0.7
aguerra	*	*	*Agentes *	Forward	0.3
mmartine	*	*	* Avance *	Reply	1.0
mmartine	*	*	* Tesis *	Reply	1.0
mmartine	*	*	* Curso *	Read	0.8
mmartine	*	*	* Curso *	Save	0.1
mmartine	*	*	* Curso *	Reply	0.1
mmartine	*	*	* Conferencia *	Read	0.8
mmartine	*	*	* Conferencia *	Save	0.1
mmartine	*	*	* Conferencia *	Reply	0.1
mmartine	*	*	* Bienvenida *	Read	0.8
mmartine	*	*	* Bienvenida *	Reply	0.2
mmartine	*	*	* Chiapas *	Read	0.7
mmartine	*	*	* Chiapas *	Save	0.2
mmartine	*	*	* Chiapas *	Forward	0.1
mmartine	*	*	* Interés *	Read	0.7
mmartine	*	*	* Interés *	Save	0.2
mmartine	*	*	* Interés *	Forward	0.1
agonzale	*	*	* Biblioteca *	Delete	0.8
agonzale	*	*	* Biblioteca *	Read	0.2
nejacome	*	*	* Hola *	Read	1.0
nejacome	*	*	* ondas *	Read	1.0
ceuribe	*	*	*	Read	0.8
ceuribe	*	*	*	Reply	0.2
e-Alert	*	*	*	Delete	1.0
psj	*	*	* Saludos *	Read	1.0
psj	*	*	* Hola *	Read	1.0
psj	*	*	* Ondas *	Read	1.0
psj	*	*	* casa *	Read	0.8
psj	*	*	* casa *	Reply	0.2
psj	*	*	* house *	Read	0.8
psj	*	*	* house*	Reply	0.2
psj	*	*	* Barbara *	Read	0.8
psj	*	*	* Barbara *	Reply	0.2
jgutierr	*	*	*	Reply	1.0
cperez	*	*	Ricardo *	Reply	1.0
cperez	*	*	* Tesis *	Reply	1.0
cperez	*	*	* Solaris*	Reply	1.0

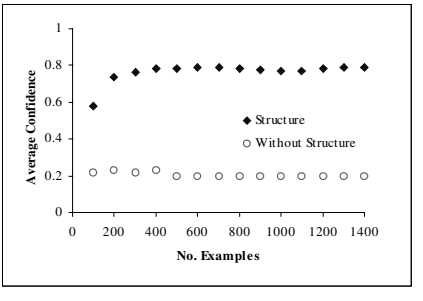
cperez	*	*	* Hola *	Read	0.7
cperez	*	*	* Hola *	Reply	0.2
cperez	*	*	* Hola *	Forwa rd	0.1
cperez	*	*	* Saludos *	Read	0.7
cperez	*	*	* Saludos *	Reply	0.3
jgonzale	*	*	* Tesis *	Save	0.7
jgonzale	*	*	* Tesis *	Reply	0.3
jgonzale	*	*	* Avance *	Save	0.7
jgonzale	*	*	* Avance *	Reply	0.3

The probability column shows conditional probabilities of the action given the attribut values, for example, the probability at the end of the first row is

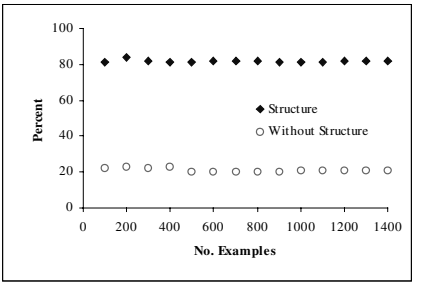
Prob(Action=read/Sender=aguerra,Day=*,Time=*,Subject=Hola)=0.90

"*" means that the attribute value is irrevelant, it is a dummy value. In our example, it means that the day of the week, and the time (hour) that the mesage is received are not relevant for the action that is taken. The above reprsentation also means that in the same situation with probability 0.10 the user will take some other action (with uniform probability) distinct from *read*. For all other attribute combinations not shown in the table, it is assumed that the user takes an action using a uniform probability distribution over the set of possible actions (these cases belong to the unstructured examples).

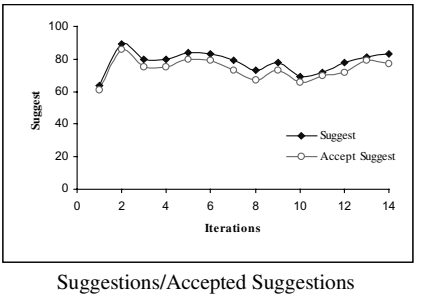
The results of the simulation are shown below.



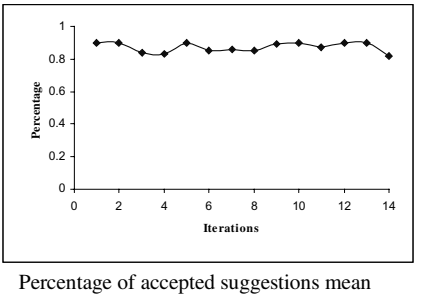
Accumulated predictive confidence



Accumulated percentage of correct predictions



Suggestions/Accepted Suggestions



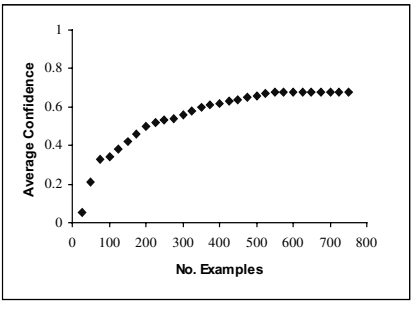
Percentage of accepted suggestions mean

As expected the agent learns very quickly the user's behavior for the structured class, and it will learn nothing within the unstructured examples class, since the actions in this case are selected randomly. Notice from the last two graphs that the overall behavior performance measured in terms of accepted suggestions is quite acceptable, being very

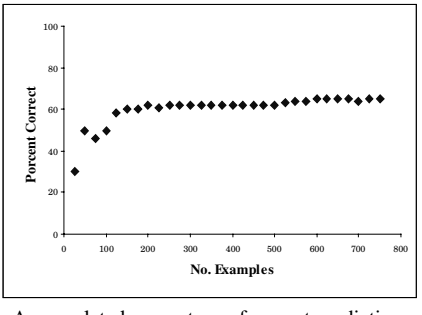
close to one. This is so because when the agent is confronted with unstructered examples it will make no suggestion since no matter how many of these are stored in memory, the predicted confidence of them will seldom exceed the threshold.

5.2 Real User

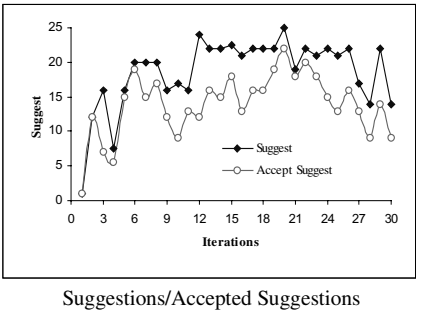
We used files generated during several months for two real users and then aply on them Hermes-II. The results obtained for one of this users are summarized in the following graphs.



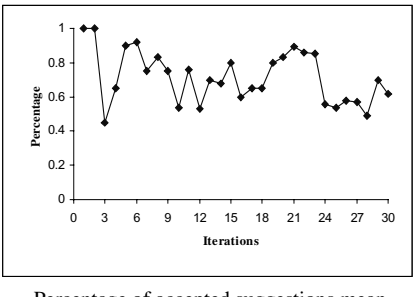
Accumulated predictive confidence



Accumulated percentage of correct predictions



Suggestions/Accepted Suggestions



Percentage of accepted suggestions mean

As can be noted the performance measures converge more slowly than in the simulated users case. This is so since a real user shows less regularities in his behavior than a simulated one.

The graphs show that confidence prediction increases with memory size, reaching a value of 0.7 for 450 examples. Percentage of correct predictions shows a similar behavior. The mean of accepted suggestions is very low at the beginning, but from the 10-th iteration on it starts to be in an acceptable level. From this point on the algorithm begins detection of regularities in the user's behavior.

From this results we can point out the following conclusions:

- MBR algorithms prove to be useful in adaptive agents
- It is feasible to use MBR with value difference metric to make interface agents learn from users behavior.

- In general MBR may be quite useful when applied to systems that generate huge data bases
- A limitation of MBR as here implemented is that it assumes conditional independence of attributes. It is possible to incorporate to the agent's structure a preprocessing module that "discovers" the agent's behavior structure , for example through the use of bayesian networks learning algorithms [Sucar and Martínez 98], which will make the learning algorithm more efficient.
- Another limitation of Hermes-II is its inability to forget, improvement in the agent's performance in the long run may be achieved by adding a forgetness module to Hermes-II.
- Hermes-II architecture, and the learning MBR algorithm are not restricted to interface agents, we can think of many other applications which involve classification and prediction, such as medical or educational applications.
- If the number of attributes is large, as memory size increases the algorithm may turn out to be quite slow; a possible solution is to think in parallel processing.

6. Bibliography

[Aguera 99] Aguera, A. S. *Razonamiento Basado en la Memoria y Comportamiento Adaptivo de Agentes*, Tesis para obtener el grado de Maestra en Inteligencia Artificial, Universidad Veracruzana, 1999.

[Aguera and Guerra 98] Aguera, A.S., and Guerra A. *Aprendizaje en Agentes Autonomos*, XIV Seminario Internacional de Ingeniería y Ciencias Básicas, Apizaco, Tlax. México,1998.

[Aguirre 98] Aguirre, A.E., *Bayes Versión 1.0*, Manual del Usuario, Maestría en Inteligencia Artificial, México, 1998.

[Boone 98] Boone, G., *Concept Features in Re: Agent, an Intelligent Email Agent*, to appear in the Second International Conference on Autonomous Agents (Agents 98), Minneapolis/ St Paul, May 10-13, 1998.

[Kaelbling 93] Kaelbling, L.P., *Learning in Embedded Systems*, Cambridge Masachussets: The MIT Press., 1993.

[Kasif et al 97] Kasif S, Salzberg S., WaltzD., Rachlin J., Aha D., *Towards a Framework for Memory-Based Reasoning*, 1997.

[Laurel 90] Laurel B., *Interface Agents: methaphors with character*, in: The Art Human Computer Interface Design, editor Brenda Laurel, Ed. Addison-Wesley, United States, 1990.

[Lieberman 97] Lieberman H., *Autonomous Interface Agents*, MIT Media Lab, 1997.

[Maes and Kozierok 93] Maes P., and Kozierok R., *Learning Interface Agents*, in: Proc. Of the Eleventh National Conf. On Artificial Intelligence, 459-465, Whashington, D.C., 1993, MIT Press.

[Maes 94] Maes P., *Agents that Reduce Work and Information Overload*, Communications of the ACM, 37(7), 31-40, 1994.

[Mitchell et al, 94] Mitchell T., Caruana R., Freitag D., and Zabowski D., *Experience with a Learning Personal Assistan*t, Communications of the ACM, 37 (7), 81-91, 1994.

[Nwana 96]Nwana H.S., *Software Agents: An Overview*, Knolowdge Enginnering Review, 11 (3), 205-244, 1996.

[Stanfill and Waltz 86] Stanfill C. and Waltz D, *Toward Memory-Based Reasoning*, Communications of the ACM, vol 29, num 8, December 1986.

A Model for Combination of External and Internal Stimuli in the Action Selection of an Autonomous Agent

Pedro Pablo González Pérez[1*], José Negrete Martínez[2], Ariel Barreiro García[3], and Carlos Gershenson García[4]

[1]Instituto de Investigaciones Biomédicas/UNAM
Instituto de Química/UNAM
ppgp@servidor.unam.mx

[2]Instituto de Investigaciones Biomédicas/UNAM
Maestría en Inteligencia Artificial/UV
jnegrete@mia.uv.mx

[3]Centro de Cibernética Aplicada a la Medicina/ISCM-H
ariel@cecam.cu

[4]Instituto de Química/UNAM
Fundación Arturo Rosenblueth
Facultad de Filosofía y Letras/UNAM
carlos@jlagunez.iquimica.unam.mx

Abstract. This paper proposes a model for combination of external and internal stimuli for the action selection in an autonomous agent, based in an action selection mechanism previously proposed by the authors. This combination model includes additive and multiplicative elements, which allows to incorporate new properties, which enhance the action selection. A given parameter α, which is part of the proposed model, allows to regulate the degree of dependence of the observed external behaviour from the internal states of the entity.

1 Introduction

The theory of behaviour-based systems (Brooks, 1986; Brooks, 1989) provides a new philosophy for the construction of autonomous agents, inspired in the ethology field. Unlike the knowledge-based systems, behaviour-based systems interact directly with their problem domain. An autonomous agent perceives its problem domain through its sensors and acts on this through its actuators. The problem domain of an autonomous agent is commonly a dynamic, complex and unpredictable environment, in which it deals to satisfy a set of goals or motivations, which could vary in time. An autonomous agent decides itself how to relate its external and internal stimuli to its motor actions in such a way that its goals can be satisfied (Maes, 1994).

* To whom all correspondence should b e sent: Laboratorio de Química Teórica. Instituto de Química. Circuito Universitario, Coyoacán, México C.P. 04510 tel. (52) 5 622 4424

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 621—633, 2000.

The term "agent" has been widely used in a great variety of research domains, being the most common those domains belonging to artificial intelligence and computer science areas. Several have been the definitions and interpretations contributed for the term "agent", depending these on the domain in which the agent has been used, as well as of the particular use that it has been given.

According to Maes (Maes, 1994) " an agent is a system that tries to satisfy a set of goals in a dynamic and complex environment. An agent is situated in the environment: it can sense the environment through its sensors and act upon the environment using its actuators".

An agent has goals, and in dependency of the domain in which the agent has been defined, these can take different forms. In order to satisfy its goals, an autonomous agent must select, at every moment in time, the most appropriate action among all the possible actions that it could execute. An action selection mechanism (ASM) is a computational mechanism, which must produce a selected action as output when different external or internal stimuli have been provided as inputs. That is, an ASM decides how to combine external and internal stimuli to select which action must be executed by the agent.

In this paper we propose a model for combination of external and internal stimuli, from an ASM developed by the authors (González, 1999), for the action selection in an autonomous agent, which exhibits some of the properties not satisfied or not explained by the ASMs that will be reviewed here. The proposed model incorporates new properties derived from the principles that characterize the animal behaviour, which must enrich the action selection.

The ASM has been structured in a distributed blackboard architecture, which, given its great capacity for the coordination and integration of several tasks in real time and its extreme flexibility for the incorporation of new functionality, facilitates the model implementation. As well as, the incremental incorporation of new properties and processes of learning directed to enrich the selection of actions, making it more adaptive.

The paper is structured as follows. The next section briefly describes some of main ASMs proposed in the literature and a comparison between these ASMs is done. This comparison is focussed to the combination scheme utilized. Section 3 presents the ASM developed by the authors, the Internal Behaviour Network (González, 1999). Here are discussed the essential characteristics of the proposed ASM that allows to understand the combination scheme that will be presented in section 4. Section 5 presents the simulation and experiments developed to verify the properties of the combination scheme.

2 Action Selection Mechanisms

The following are examples of ASMs and related works to action selection:

- Tinbergen's mechanism (Tinbergen, 1950; Tinbergen, 1951), a hierarchic network of nodes or centres, which approaches the complete action selection problem with a noticeable emphasis in the reproductive stage.
- Lorenz's psycho-hydraulic mechanism (Lorenz, 1950; Lorenz, 1981), a model that tries to explain some ethology phenomena, without approaching the action selection problem complete.

- Baerends' model (Baerends, 1976), a hierarchic network of nodes, a model inspired by ethologist studies made in particular classes of insects and birds.
- Brooks' subsumption architecture (Brooks, 1986; Brooks, 1989), which can be used as a mean to implement robot control systems, which include tasks of perception and action, in addition to the emergency of behaviours
- Rosenblatt and Payton's hierarchical network (Rosenblatt and Payton, 1989), a mechanism very similar in many aspects to the hierarchical models proposed by Tinbergen and Baerends, but with nodes very similar to formal neurons.
- Maes' bottom-up mechanism (Maes, 1990; Maes, 1991), a distributed non-hierarchical network of nodes, where each node represents an appetitive or consummatory alternative that the entity can execute.
- Beer's neural model (Beer, 1990; Beer, Chiel and Sterling, 1990), a semi-hierarchical network of nodes, where each node represents a neuronal circuit.
- Halperin's neuroconnector network (Hallam, Halperin and Hallam, 1994), a non-supervised neural network organized in layers.
- Negrete's neuro-humoral model (Negrete and Martinez, 1996), a non-hierarchical distributed network of nodes, where each node is a neuro-humoral neuron.
- Goetz's recurrent behaviour network (Goetz and Walters, 1997), a network of nodes, where a node can represent a behaviour, a sensor or a goal; the network converges to a particular behaviour (attractor), in a similar way that a Hopfield's network (Hopfield, 1982) converges to a certain pattern.

Table 1 shows a comparison between different ASMs before mentioned, taking into account the most relevant aspects from these. In particular, our interest is focussed to the form in which these mechanisms combine the external and internal stimuli to select an external action.

The great majority of these action selection mechanisms propose an additive model for stimuli combination, as showed in Table 1. The stimuli addition could be interpreted in many ways, but the essence is that this type of stimuli combination models a behaviour with a more reactive than a motivated tendency. This can be explained in the following terms: for high values of the external inputs, the additive combination could surpass some value threshold previously established and, therefore, shoot a certain external action, still for non significant values of the internal inputs.

On the other hand, in a pure model of multiplicative combination both types of inputs are necessary so that an external consummatory action of the entity can be executed. This is, if we considered that $\Sigma_j S_j$ is the sum of all the associated external inputs to the internal state E_i, then $\Sigma_j S_j * E_i \neq 0$ only if $\Sigma_j S_j \neq 0$ and $E_i \neq 0$. Pure multiplicative models have a limit ratio to very small values of the external inputs and/or of the internal inputs, since it does not matter how great is the external input if the internal input is near zero and vice versa. A more detailed discussion of the advantages and disadvantages of the ASMs here mentioned can be found in (González, 1999).

An alternative to eliminate the undesired effects of the pure additives and multiplicative combinations is to consider a model that incorporates the most valuable elements of both types of combination. In this paper, we propose a model of external and internal inputs combination for the action selection, where both types of combinations

Table 1.

ASM	Disciplines	Architecture	Combination of stimuli	Learning schemes
Tinbergen	ethology	hierarchical network of nodes, where each node represents a kind of behaviour	summed	none
Lorenz	ethology, psychology and hydraulic engineering	psycho-hydraulic model	summed	none
Baerends	ethology	hierarchical network of nodes, where each node represents a kind of behaviour	unstated	none
Brooks	robotic	distributed network of finite state machines	unstated	
Rosenblatt and Payton	robotic and artificial neural networks	connectionist, feed-forward network, behaviours are defined by connections between processing elements	can be any function of weighted inputs	none
Maes	ethology and behaviour-based systems	non-hierarchical, distributed network, where each node represents a kind of behaviour	summed	none
Beer	ethology, neuroethology and artificial neural network	semi-hierarchical network, where each node is a neural network implementing a particular kind of behaviour	summed	none
Halperin	ethology and artificial neural network	non-supervised, hierarchical, feed-forward network	summed	classical, secondary, and postponed conditioning
Negrete	neurophysiology ethology	non-hierarchical, distributed network of neuro-humoral neurons	summed	none
Goetz	artificial neural network and attractors theory	recurrent distributed network	summed	none

take place. This is, in the proposed model the external and internal inputs interact multiplicatively, which can be interpreted considering that both factors are necessary to activate an external consummatory action. Nevertheless, the model also contains additive elements, which allow to give a greater importance or weight to the internal inputs, and are finally those that determine that interesting properties as the action selection directed towards internal medium of the entity, and the execution of an external behaviour oriented to the search of a specific signal can emerge product from this type of combination.

3 The Internal Behaviour Network

The ASM proposed by us has been structured from a network of blackboard nodes developed by the authors (González and Negrete, 1997; Negrete and González, 1998). A blackboard node is integrated by the following components: a set of independent modules called knowledge sources, which have specific knowledge about the problem domain; the blackboard, a shared data structure through which the knowledge sources communicate to each other by means of the creation of solution elements on the blackboard; the communication mechanisms, which establish the interface between the nodes and a node and the external and internal mediums; the activation state registers of the knowledge sources (REAC); and a control mechanism, which determines the order in that the knowledge sources will operate on the blackboard.

As it can be appreciated in Figure 1, the actual architecture of the ASM exhibits two blackboard nodes: the cognitive node and the motivational node. This architecture has been called Internal Behaviour Network (IBeNet) (González, 1999). The term " internal behaviour " has been used to describe the processes of creation and modification of solution elements that occur on the blackboard of the node. Structurally, an internal behaviour is a package of production rules (if <condition> then <action>). A production rule is also known as an elementary behaviour. The condition of an elementary behaviour describes the configuration of solution elements on the blackboard that is necessary, so that the elementary behaviour contributes to the solution processes of the problem. The way in which an elementary behaviour contributes to the solution of the problem is specified in its action, which can consist of the creation or modification of solution elements in certain blackboard levels.

The cognitive node blackboard is structured in six levels of abstraction: external perceptions, perceptual persistents, consummatory preferents, drive/perception congruents, potential actions, and actions. For the cognitive node, the following internal behaviours have been defined: perceptual persistence, attention to preferences, reflex response inhibition, and external behaviours selector. The defined mechanisms of communication for this node are the exteroceptors and actuators, which establish the interface between the cognitive node and external medium; and the receptor and transmitter mechanisms, which establish communication with the motivational node.

The role of the cognitive node includes the processes of representation of the perceptual signals, integration of internal and external signals, inhibition of reflex responses, and the selection of the external behaviour that adjusts better to the actual external conditions and internal needs.

The blackboard of the motivational node is structured in four abstraction levels: internal perceptions, external perceptions, propio/extero/drive congruents, and drive. The internal behaviours that operate in the motivational node are the following: propio/extero/drive congruence and consummatory preferences selector. The communication of this node is established from the propioceptors, which define the interface between the node and the internal medium; and the receptor and transmitter mechanisms, which establish the communication with the cognitive node.

The role of the motivational node includes the processes of representation of internal signals, combination of internal and external signals and the selection of the more appropriate consummatory preference.

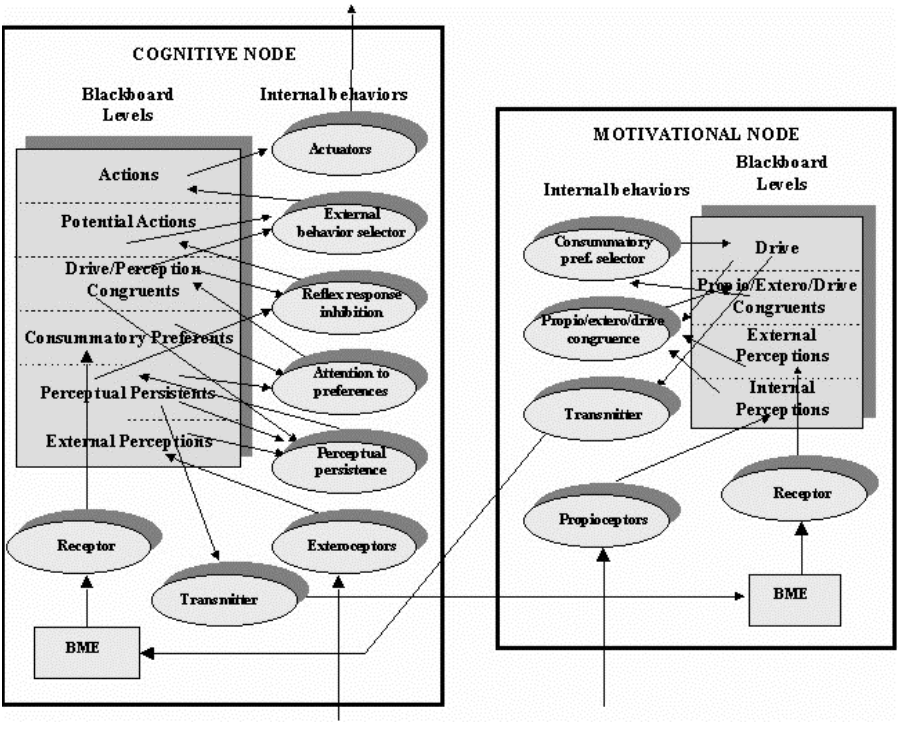


Fig. 1 Structure of the blackboard node network. Here have been omitted the activity state registers.

A full presentation of the structure and properties of the IBeNet can be found in (González, 1999).

4 The Model fo External and Internal Stimuli Combination

At the level of the propio/extero/drive congruence internal behaviour of the motivational node is carried out the external and internal signals combination to determine the action that must be executed by the entity. These signals are registered in the external perceptions and internal perceptions levels, respectively, of the motivational node. The solution elements registered in the external perceptions level are originated by unconditional stimuli (US), conditional stimuli (CS), or both; all of them initially projected in the external perceptions level and later recovered from the perceptual persistents level, of the cognitive node. On the other hand, the solution elements registered in the internal perceptions level of the motivational node represent the values of the internal states that are able to satisfy part of the conditions of certain propio/extero/drive congruence elementary behaviours that operate on this level. The scheme of signals combination used by this internal behaviour is the one given by the expression (1).

$$A_i^C = O_i^E * (\alpha + \Sigma_j Fa_{ij}^S * O_j^S) + O_i^D \quad (1)$$

where: A_i^C is the certainty value with which will be created the element solution C_i in the propio/extero/drive congruents level of the blackboard, α is the weight or importance attributable to the internal state ($0 \leq \alpha \leq 1$), O_i^E is the internal signal, O_i^D is the signal created in the Drive level ($O_i^D = 0$, for all i at the initial moment $t = t_0$), O_j^S are the external signals associated to the internal state O_i^E, and Fa_{ij}^S are the coupling strengths of the elemental behaviours propio/extero/drive congruence (coupling strengths are to elemental behaviours what synaptic weights are to artificial neural networks).

The propio/extero/drive congruence internal behaviour does not require the creation of REACs. Whenever the condition of an elementary behaviour is satisfied, this executes its final action, creating the solution element C_i with certainty A_i^C in the propio/extero/drive congruents level of the blackboard of the motivational node. The level of certainty A_i^C represents the activity of the elementary behaviour i. Figure 2 shows the structure of an elementary behaviour of this type.

Parameters:
condition solution elements: E_i, S_j, D_i (j = 1, 2, ...)
action solution elements: C_i
coupling strengths: Fa_{ij}^S (j = 1, 2, ...)
importance attributable to the internal state: _
Condition:
(_ = 0 AND
blackboard-level = Internal Perceptions AND
cert(E_i, O_i^E) AND
O_i^E _ 0 AND
blackboard-level = External Perceptions AND
cert(S_i, O_i^S) AND
O_i^S _ 0) OR
(___ 0 AND
blackboard-level = Internal Perceptions AND
cert(E_i, O_i^E) AND
O_i^E _ 0 AND
blackboard-level = External Perceptions AND
cert(S_i, O_i^S))
Action:
calculate $A_i^C = O_i^E * (\_ + \__i Fa_{ii}^S * O_i^S) + O_i^D$ AND
blackboard-level = Propio/Extero/Drive Congruents AND
create-solution-element (C_i, A_i^C)

Fig. 2 Structure of a propio/extero/drive congruence elementary behaviour.

The role of the propio/extero/drive congruence elementary behaviour are next described for the cases when $\alpha = 1$ and $\alpha \neq 0$.

Case 1: α=0

When α is equal to zero, the propio/extero/drive congruence elementary behaviour will be able to activate only when the internal and external signals ($O_i^E \neq 0$ and $\Sigma_j Fa_{ij}^S * O_j^S \neq 0$) associated to this coincide. If the necessary internal state is not present, then the external signal is not sufficient by itself to evoke the motor action, since the attention to preferences internal behaviour is activated only when their external signal (solution element in the perceptual persistent level of the cognitive node) and the drive signal (solution element in the consummatory preferents of the cognitive node) coincide. On the other hand, although a very strong internal signal exists, corresponding to an urgent necessity of the entity, if the usable external signal is not present, the propio/extero/drive congruence elementary behaviour will not activate. Then, when $\alpha = 0$, both factors (external signal and internal signal) are necessary so that the propio/extero/drive congruence internal behaviour activates. Physiologically this can be interpreted, when thinking that the inputs that come from the internal states sensitize this internal behaviour with the signals that come from the external medium.

Case 2: $\alpha \neq 0$

When α is different of zero, then there is being granted a little more weight (or importance) to the internal state than to the external inputs. So that, still in total absence of external inputs, the propio/extero/drive congruence elementary behaviour could be activated for a very strong value of the internal signals (when the value of α is near one). Thus, when the internal necessity is extremely high and external signals do not exist, the propio/extero/drive congruence elementary behaviour can be activated, and therefore to cause a type of "preactivation" of the conditions of the attention to preferences elementary behaviours (in particular, only one of these elementary behaviours will be the one that will receive greater preactivation). This it is a typical example of an internal behaviour whose actions are directed to the internal medium. This mechanism constitutes the underlying base of the exploratory behaviour oriented to a specific objective, which is another of the distinguishing properties of the IBeNet. When there is a total satiety of the internal necessity that caused the activation of the propio/extero/drive congruence elementary behaviour ($O_i^E = 0$), then the level of activation of this elementary behaviour will decrease to zero ($A_i^C = 0$), independently of the existence of usable external signals. For activation levels of zero or very near to zero the elementary behaviour will not be activated. Of this way, the observed external behaviour always will be adaptive.

5 Simulation and Experiments

5.1 The simulation

In order to verify the role of the parameter α in the combination of external and internal inputs, a program was written implementing the IbeNet, and used by an autonomous mobile robot (animat) simulated in virtual reality. The simulated robot was inspired by an original idea developed by Negrete (Negrete and Martinez, 1996). The graphical representation of the animat is a cone with a semicircumscript sphere. This has primitive

motor and perceptual systems developed specially for this simulation. The animat has internal states to satisfy, such as hunger, thirst and fatigue, and the qualities of strength and lucidity; which are affected by the diverse behaviours that this can execute. The implemented behaviours were: wander, explore, approach a specific signal, avoid obstacles, rest, eat, drink, and runaway. An environment was also simulated, in which the animat develops. In this environment, obstacles, sources of food and water, grass, blobs (aversive stimuli), and spots of different magnitudes can be created, which animat can use to satisfy its internal necessities. This simulation can be accessed via Internet in the address http://132.248.11.4/~carlos/asia/animat/animat.html .

5.2 The experiments

The developed experiments using this simulation were directed to verify: (1) the influence of the internal states in the observed external behaviour of the animat, (2) the role of the competition at a motivational level in the selection of the external behaviour to execute, (3) the exploratory behaviour oriented to the search of a specific signal and the reduction of the response times of the animat, (4) the stability in the selection and persistence in the execution of the external behaviours, (5) the discrimination between different stimuli taking in count the quality of them, (6) avoid aversive stimuli, (7) the non-persistence in the execution of a consummatory action when an aversive stimulus is perceived, and (8) the role of the learning processes in the action selection. Here, experiment (3) is illustrated, which was oriented to the verification of the following hypothesis:

"The value of α in the expression (1) is closely related with the time of response of the animat between the presentation of an external stimulus, for which a very high internal necessity exists, and the consummation of the action associated to the satisfaction of this necessity, being this an inversely proportional relation".

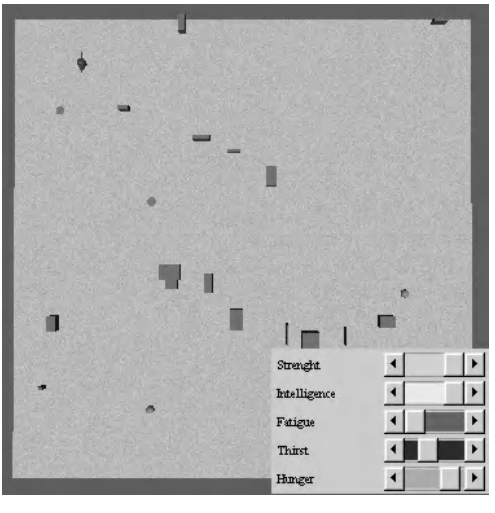


Fig. 3. Initial state: water perceived, food not perceived, little thirst and much hunger.

That is, if the value of α increases, then the time of reaction decreases; and if the value of α diminishes, then the time of reaction increases.

In order to demonstrate this hypothesis, we have created a variable that quantifies the time of reaction of the animat, from the moment at which the stimulus in the external medium appeared until the moment at which the corresponding action is executed. The value of this variable, which we will denote like RTIME, is calculated in the following way: given an external stimulus O_i^+, for which a very high internal necessity exists, the value of RTIME is equal to the amount of passed cycles of execution from the moment at which the O_i^+ stimulus is presented, until the moment at which the corresponding action is executed.

Now let us consider a simulated environment as the one shown in Figure 3, in which the stimulus "food source" is not perceived; whereas several stimuli "water source" can be perceived. Let us consider in addition that exists an urgent "eat" need, whereas the "drink" need is not relevant, as it is reflected in the values of the internal states in this same figure. Having in count these considerations, we analyse now which will be the action executed by animat in each one of the cases $\alpha = 0{,}0$ and $\alpha \approx 1.0$.

Case 1: $\alpha = 0$

Let us remember that when α is equal to zero, a propio/extero/drive congruence elementary behaviour will be able to be activated only when the external and internal signals associated to it agree. Then, in this case propio/extero/drive congruence elementary behaviour associated to thirst internal state will be activated, since for this there is an usable external input and an internal state, that although is not high, it is different from zero; whereas the propio/extero/drive congruence elementary behaviour associated to the hunger internal state cannot be activated, since although the value of this internal state very is elevated, there is no external signal able to satisfy this (we remember that the food has not been perceived).

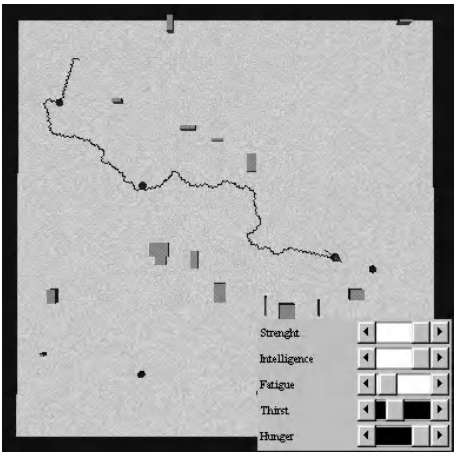


Fig. 4 Trajectory and actions executed by the animat, considering the shown initial state in 3, and $\alpha = 0$.

By virtue of the previous, and as shown in Figure 4, what has happened is that the propio/extero/drive congruence elementary behaviour associated to the thirst internal input prevailed at a motivational level, reason why the drink external behaviour was executed, and the animat has lost time in finding the food, whereas the hunger internal state continues to increase.

Case 2: $\alpha \approx 1$

Here we must consider that when $\alpha \approx 1$, to the internal state associated to a propio/extero/drive congruence elementary behaviour is being granted with a little more weight than to the external inputs associated to this, so that, still in total absence of external inputs, this elementary behaviour could be activated for a very strong value of the internal state. Therefore, in this case, both the propio/extero/drive congruence elementary behaviour associated to the thirst internal state, and the one associated to the hunger internal state will be activated, reason why in the competitive process that is carried out to motivational level both elementary behaviours will participate.

By virtue of the previous, and as it can be seen in Figure 5, for a very strong value of α (near 1), the propio/extero/drive congruence elementary behaviour associated to the hunger internal input was winner in the competition, given the low value of the thirst internal input. This fact caused the "preactivation" of the conditions of the attention to preferences elementary behaviours that operate on the blackboard of the cognitive node, with none of these being activated. Because of the previous, the external behaviour exploration oriented to the search of a necessary external signal was activated, being in this case "food source" such signal.

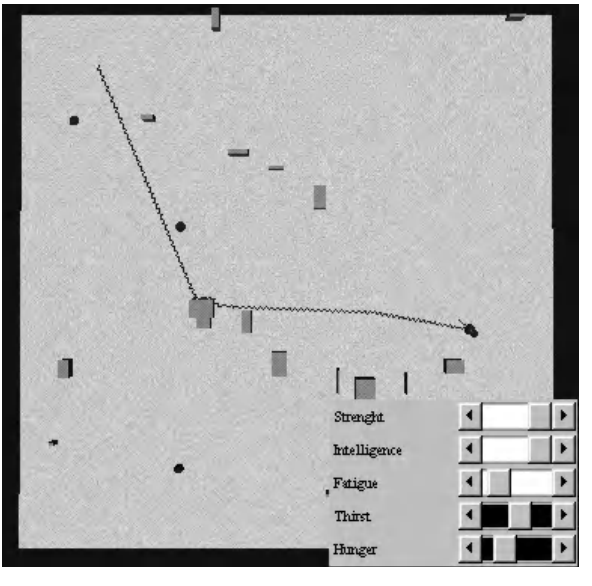


Fig. 5. Trajectory and actions executed by animat, considering the initial state shown in Figure 3, and $\alpha \approx 1$.

Furthermore, when the exploratory behaviour oriented to the search of a specific signal is being generated, the animat has a higher probability of finding the source of food faster that in the previous case, when α was equal to 0; which is more adaptive, since the animat's urgent necessity is to eat. To this, it is necessary to add that the response time between the moment at which the food source is perceived directly and the moment in which the eat behaviour is executed will be much smaller, although there are other internal inputs present.

6 Conclusions

One of the main results derived from the proposed combination scheme of external and internal inputs, has been to understand that the essential difference between the diverse ASMs reviewed goes beyond the type of architecture in which these have been structured, or of the model by which these have been inspired. The essential difference is in the way in which these ASMs combine the external and internal stimuli to select a determined action, and therefore, in the diverse repertoire of properties that characterizes this combination.

In this way, some of the properties observed in the developed actions selection mechanism are derived directly from the value of α in the combination scheme. Examples of these are: the strong dependency of the observed external behaviour of the internal states of the entity, and the actions selection towards external medium and towards the internal medium of the entity. Among other properties of interest exhibited by the mechanism, there are: the stability in the selection of actions, the persistence in the execution of an action, the existence of an external behaviour oriented to the search of a specific signal, and the explicit relation of the selection of actions with the learning processes.

Based on the characteristics already exposed, the degree of dependency of the observed external behaviour of the internal states of the entity can be regulated with the parameter α . Of this way, when α is near zero, the observed behaviour will have a tendency to the reactivity; whereas when α is near one, the observed behaviour will be motivated. Depending on the problem domain in which the agent is implementing the IBeNet, the parameter α grants flexibility to determine the type of relation to establish between the internal and external inputs, giving a greater or smaller importance to these.

Following this idea, considering a dynamic value of α, which is learned by the agent, and establishing different values of α for each behaviour; would allow the agent to adapt in order to decide in what situations it must behave in a more motivated or more reactive way. The development of this idea is part of the future work of the authors.

References

Baerends, G. (1976) The functional organization of behaviour. *Animal Behaviour*, 24, 726-735.

Beer, R. (1990) Intelligence as Adaptive Behaviour: an Experiment in Computational Neuroethology. Academic Press.

Beer, R., Chiel, H. and Sterling, L. (1990) A biological perspective on autonomous agent design. *Robotics and Autonomous Systems*, 6, 169-186.

Brooks, R. A. (1986) A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation.* RA-2, April, pp. 14-23.

Brooks, R. A. (1989) A robot that walks: Emergent behaviour from a carefully evolved network. *Neural Computation*, 1, 253-262.

Goetz, P. and D. Walters (1997) The dynamics of recurrent behaviour networks. *Adaptive Behaviour*, 6(2), 245-282.

González, P.P. (1999) Redes de Conductas Internas como Nodos-Pizarrón: Selección de Acciones y Aprendizaje en un Robot Reactivo. *Tesis Doctoral*, Instituto de Investigaciones Biomédicas/UNAM, México.

González, P.P. and J. Negrete (1997) REDSIEX: A cooperative network of expert systems with blackboard architectures. *Expert Systems*, 14(4), 180-189.

Hallam, B.E, J. Halperin and J. Hallam (1994) An Ethological Model for Implementation in Mobile Robots. En *Adaptive Behaviour*, 3 (1), pp 51-79.

Hopfield, J.J. (1982) Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79, 2554-2558.

Lorenz, K. (1950) The comparative method in studying innate behaviour patterns. Symposia of the Society for Experimental Biology, 4, 221-268.

Lorenz, K. (1981) Foundations of Ethology. Springer-Verlag.

Maes, P. (1990) Situated agents can have goals. *Journal of Robotics and Autonomous Systems*, 6(1&2).

Maes, P. (1991) A bottom-up mechanism for behaviour selection in an artificial creature. En J.A. Meyer and S.W. Wilson (ed.), *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behaviour* MIT Press/Bradford Books.

Maes, P. (1994) Modeling Adaptive Autonomous Agents. *Journal of Artificial Life*, 1(1,2), MIT Press.

McFarland, D.J. and A.I. Houston (1981) Quantitative Ethology: The state space approach. Boston: Pitman.

Negrete, J. and M. Martínez (1996) Robotic Simulation in Ethology. *Proceedings of the IASTED International Conference: Robotics and Manufacturing*, Honolulu, Hawaii, USA, pp. 271-274.

Negrete, J. and P.P. González (1998) Net of multi-agent expert systems with emergent control. *Expert Systems with Applications*, 14(1) 109-116.

Pfeifer, R. and C. Scheier (1999) Understanding Intelligence. MIT Press.

Rosenblatt, K. and D. Payton (1989) A fine-grained alternative to the subsumption architecture for mobile robot control. *Proceedings of the IEEE/INNS International Joint Conference on Neural Networks,* IEEE.

Tinbergen, N. (1950) The hierarchical organization of mechanisms underlying instinctive behaviour. *Experimental Biology*, 4, 305-312.

Tinbergen, N. (1951) The Study of Instinct. Claredon Press.

Action Selection Properties in a Software Simulated Agent

Carlos Gershenson García*[1], Pedro Pablo González Pérez[2], and José Negrete Martínez[3]

[1] Instituto de Química/UNAM
Fundación Arturo Rosenblueth
Facultad de Filosofía y Letras/UNAM
carlos@jlagunez.iquimica.unam.mx
[2] Instituto de Investigaciones Biomédicas/UNAM
Instituto de Química/UNAM
ppgp@servidor.unam.mx
[3] Instituto de Investigaciones Biomédicas/UNAM
Maestría en Inteligencia Artificial/UV
jnegrete@mia.uv.mx

Abstract. This article analyses the properties of the Internal Behaviour network, an action selection mechanism previously proposed by the authors, with the aid of a simulation developed for such ends. A brief review of the Internal Behaviour network is followed by the explanation of the implementation of the simulation. Then, experiments are presented and discussed analysing the properties of the action selection in the proposed model.

1 Introduction

By the middle 1980's, researchers in the areas of artificial intelligence (AI), computer sciences, cognitive sciences and psychology realized that the idea of computers as intelligent machines was inappropriate. Inappropriate because the brain functions, far from being related with mathematical proofs and programmes execution, are related with the control of behaviour. Most researchers now agree that intelligence is manifested in behaviour (Pfeifer and Scheier, 1999). This has given place a research area to develop known as Behaviour-based Systems (BBS) (Brooks, 1986), in order to model intelligence in a different approach than previous Knowledge-based Systems (KBS).

This new line of research, also known as "autonomous agents", was inspired mainly in ethology, the branch of biology that studies animal behaviour. The term "agent" has been used in many other areas, but in this paper, when we talk about autonomous agents, we refer to the ones proposed by BBS. In BBS, the interaction with the problem domain is direct, while in KBS it is more limited. An autonomous agent perceives its problem

* To whom all correspondence should b e sent: Laboratorio de Química Teórica. Instituto de Química. Circuito Universitario, Coyoacán, México C.P. 04510 tel. (52) 5 622 4424

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 634—648, 2000.

domain through its sensors and actuates over it through its actuators. The problem domain of an autonomous agent is commonly a dynamic, complex and unpredictable environment, in which the agent tries to satisfy a set of goals or motivations, which can vary in time. An autonomous agent decides by himself how to relate his external and internal inputs with his motor actions in such a way that its goals may be achieved (Maes, 1994).

To satisfy his goals, an autonomous agent must select, at every moment in time, the most appropriate action among all possible actions that he could execute. This is what, in the context of BBS, is known as the action selection problem (ASP). While the ASP refers to what action, the agent (robot, animat or artificial creature) must select every moment in time; an action selection mechanism (ASM) specifies how these actions are selected. An ASM is a computational mechanism that must produce as an output a selected action when different external and/or internal stimuli have been given as inputs. In this point of view, an ASP indicates *which*, when an ASM indicates *how*.

Among the principal ASM and related works with the action selection are the hierarchical network of centres of Tinbergen (Tinbergen, 1950; Tinbergen, 1951), the psycho-hydraulic model of Lorenz (Lorenz, 1950; Lorenz, 1981), the hierarchical nodes network of Baerends (Baerends, 1976), the subsumption architecture of Brooks (Brooks, 1986; Brooks, 1989), the connectionist model of Rosenblatt and Payton (Rosenblatt and Payton, 1989), the bottom-up mechanism of Maes (Maes, 1990; Maes, 1991), the neuronal model of Beer (Beer, 1990; Beer, Chiel and Sterling, 1990), the neuroconnector network of Halperin (Hallam, Halperin and Hallam, 1994), the neuro-humoral model of Negrete (Negrete and Martínez, 1996), and the recurrent behaviour network of Goetz (Goetz and Walters, 1997). A complete revision of these ASMs is presented in (González, 1999), along with a comparison with the proposed ASM. These mechanisms have been inspired in models belonging to disciplines such as ethology, psychology, cognitive sciences, robotics, engineering, artificial neural networks and AI, among others. Some of these mechanisms contemplate completely the ASP, while some of them only deal with part of the problem.

In this work, the action selection properties of a proposed model by the authors (González, 1999) are illustrated in a simulation of a robot created for such effect. The proposed model was based in a distributed blackboard architecture, which, given its great capacity for coordination and integration of many tasks in real time and its extreme flexibility for the incorporation of new functionality, eases the implementation of the model and the incremental incorporation of new properties and learning processes, which enrich the action selection, making it more adaptive.

Some of the properties observed in the developed ASM are: (1) the strong dependence of the observed external behaviour in the internal states of the entity, (2) the action selection to the extern medium and to the internal medium, (3) the stability in the action selection, (4) the persistence in the execution of an action, (5) the existence of an external behaviour oriented to the search of a specific signal, and (6) the explicit relationship between the action selection and the learning processes.

This article is structured as follows: in the next section the main structural and functional characteristics of the ASM that we have proposed are presented: the internal behaviour network built with blackboard nodes (González, 1999). In section 3, the developed simulation is described: the animat, the simulated environment, and the

behaviours that can be executed by the entity. Finally, section 4 presents and discusses some of the experiments developed in order to verify when the internal behaviour network was able to produce the effects claimed by it.

2 The Internal Behaviour Network

We have named "internal behaviour network" (IBeNet) to the action selection mechanism we have developed. A blackboard node is a blackboard system (Nii, 1989) with elements defined in the REDSIEX (González and Negrete, 1997) and ECN-MAES (Negrete and González, 1998) architectures. Although the blackboard architecture was developed in the area of KBS, its properties suit BBS as well. A complete explanation of the functionality of the IBeNet can be found in (González, 1999). The IBeNet architecture can be seen in Figure 1.

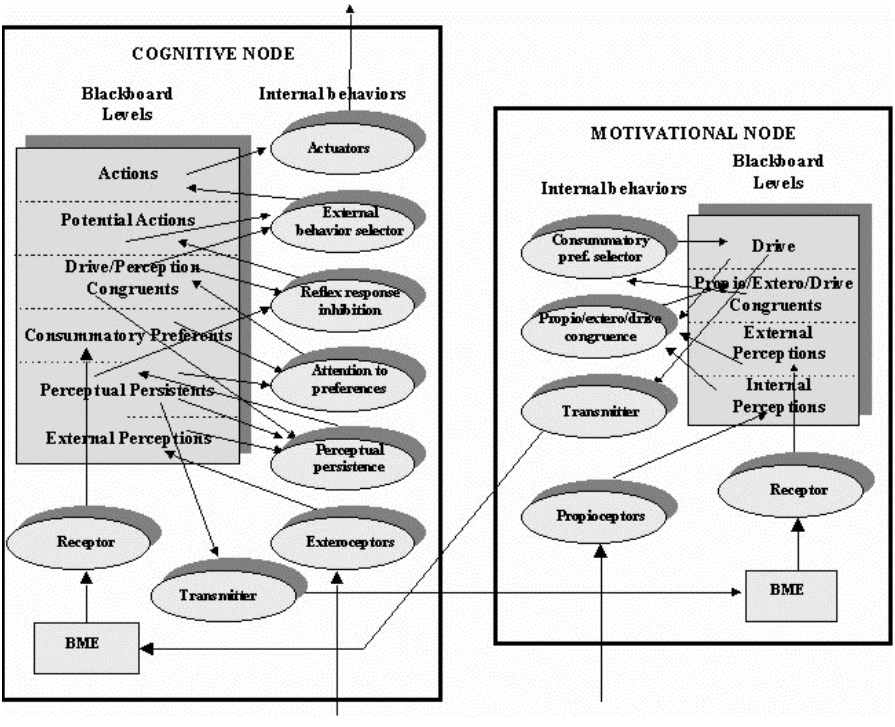


Fig. 1 Structure of the blackboard node network. Here have been omitted the activity state registers.

As shown in Figure 1, the actual structure of the IBeNet defines two blackboard nodes: the cognitive node and the motivational node. The internal tasks required for the control of the action selection can only be satisfied over the base of the cooperative work between both nodes. The inputs of the exteroceptors come from the perceptual system, while the inputs of the propioceptors come from the internal medium. The outputs of the actuators are directed to the motor system.

2.1 Elements of a Blackboard Node

The structure of a blackboard node is defined by five basic elements: the blackboard, the internal behaviours, the activity state registers (REACs) of the internal behaviours, the interface/communication mechanisms, and the competition mechanism.

The blackboard is a shared data structure over which the internal behaviours execute their final actions, and by which the internal behaviours communicate. The interface/communication mechanisms also operate over the blackboard. The internal behaviours produce changes in the blackboard, which incrementally lead to the formation of a solution of the ASP.

The term "internal behaviour" is used to describe the intrinsic actions of the information process that occur at a blackboard node level. This is, inside the entity. In this sense, an internal behaviour may be of two kinds: the ones that generate actions directed to the extern medium of the entity, and those directed to the internal medium of the entity. From the structural point of view, an internal behaviour is defined as a package of simpler mechanisms called elemental behaviour, commonly from the same kind, and structured as production rules (if <condition> then <action>).

When the conditions of an elemental behaviour are satisfied, this is activated creating an activity state register (REAC). A REAC is a data structure that describes the characteristics of the action activated by the behaviour. When a REAC is selected by the competition mechanism, the action of the internal behaviour associated to the REAC is executed, and a new solution element will be created over the blackboard, or the certainty value of an existent solution element will be modified.

Over the blackboard the interface/communication mechanisms also operate. Three kinds of interface mechanisms have been defined: (1) the exteroceptors, which establish the interface between the perceptual system and the cognitive node; (2) the propioceptors, which establish the interface between the internal medium and the motivational node; and (3) the actuators, which define the interface between the cognitive node and the motor system. The communication mechanisms are the ones in charge of executing the reception and transmission of signals from and to other nodes in the network. (In ECN-MAES (Negrete and González, 1998) this are the behaviours with social knowledge).

A list of REACs (L-REAC) is the set defined by all the REACs associated to elemental behaviours of the same kind. To the level of each L-REAC a competitive process is established between the REACs in order to decide which elemental behaviour(s) will be winner(s). Only winner behaviours will be able to execute their final action over the blackboard. The competitive process can be of two kinds: at the end of the competition more than one behaviour may win, or only one behaviour may be proclaimed winner (the winner takes all).

2.2 The Cognitive Node

For the cognitive node, the following internal behaviours have been defined: perceptual persistence, attention to preferences, reflex response inhibition, and external behaviours selector; the interface mechanisms exteroceptors and actuators; and the communication mechanisms receptor and transmitter.

The cognitive node receives signals from the perceptual system through the exteroceptors and of the motivational node through the receptor mechanism; and sends signals to the motivational node through the transmitter mechanism and to the motor system through the actuators. The cognitive node role contains the processes of representation of perceptual signals, integration of internal and external signals, reflex response inhibition, and selection of the external behaviour that best fits to the actual internal state.

As can be seen in Figure 1, the domain blackboard of the cognitive node organizes the solution elements in six abstraction levels: external perceptions, perceptual persistents, consummatory preferents, drive/perception congruents, potential actions, and actions.

2.3 The Motivational Node

The motivational node receives signals from the internal medium through the propioceptors and from the cognitive node through the receptor mechanism; and sends signals to the cognitive node through the transmitter mechanism. It has defined the following internal behaviours: propio/extero/drive congruence and consummatory preferences selector.

The role of the motivational node contains the combination of internal and external signals, and the competition to the motivational level of motivationally incompatible behaviours. This is, the final observed external behaviour in the entity is strongly dependent of its internal states. All the internal states, for which there are useful external signals compete between them to determine the final external behaviour that the entity will execute. The competition is of the winner takes all type.

The domain blackboard of the motivational node organizes the solution elements in four abstraction levels: internal perceptions, external perceptions, propio/extero/drive congruents, and drive.

3 The Simulation

In order to verify when the IBeNet was able to produce by itself the effects claimed by it, a computer programme was written implementing this model, being used by an autonomous mobile robot (animat) simulated in virtual reality. The simulated robot was inspired in an original idea conceived by Neg rete (Negrete and Martínez, 1996).

This virtual reality simulation can be accessed via Internet at the following web address: http://132.248.11.4/~carlos/asia/animat/animat.html

3.1 The Simulated Robot

As can be seen in Figure 2, the graphical representation of the animat is a cone with a semicircumscript sphere. The tip of the cone indicates the animat's direction.

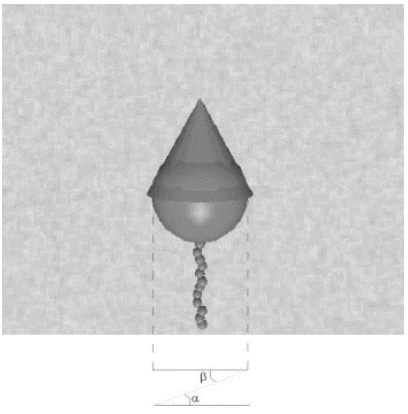


Fig. 2 Graphical representation of the animat

The internal structure of the animat can be described in terms of four basic components: the perceptual system, the internal medium, the action selection mechanism, and the motor system. In Figure 3 this components and its interrelations can be appreciated.

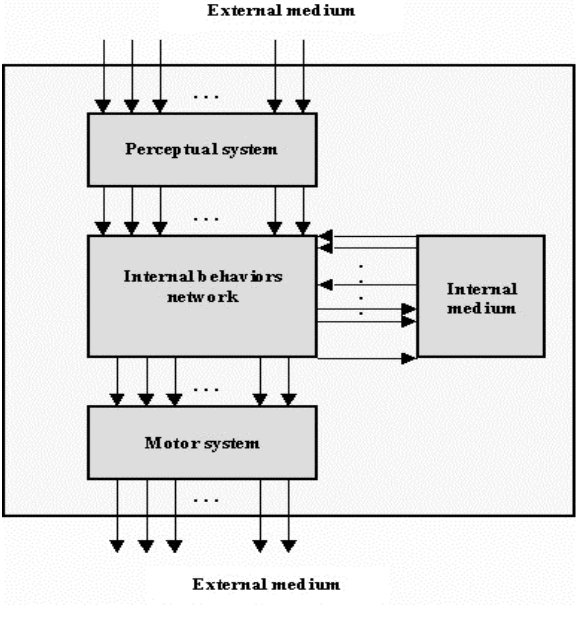


Fig. 3 Components that integrate the internal structure of the animat and its interrelations.

3.1.1 The Perceptual System

The animat is situated in a plane (z,x) of a space (x,y,z). In this plane there can be found various kinds of external stimuli: water, represented by blue circles; food, represented by

green spheres; grass, represented by texturized green circles; obstacles, represented by brown parallelepipeds; blobs (aversive stimuli), represented by black ellipsoids; and red and yellow spots (initial neutral stimuli), represented by circles of the respective colour. The animat only perceives those external stimuli that are inside the perceptual region (R_p) defined by the semicircle determined by (1).

$$R_p = \begin{cases} ((z-z_a)^2+(x-x_a)^2<r_p^2)\cap(x>x_a+\tan((th+\pi/2)*(z-z_a))) & \text{if } 0<th\leq\pi \\ ((z-z_a)^2+(x-x_a)^2<r_p^2)\cap(x<x_a+\tan((th+\pi/2)*(z-z_a))) & \text{if } \pi<th\leq 2\pi \end{cases} \quad (1)$$

The expression $(z-z_a)^2+(x-x_a)^2<r_p^2$ determines the set of all points contained in the circle with centre in (z_a,x_a) and radius r_p (r_p stands for perception radius). The expression $x>x_a+\tan((th+\pi/2)*(z-z_a))$ determines the set of all points above the straight line perpendicular to the animat's orientation (th) and which crosses the point (z_a,x_a), while the expression $x<x_a+\tan((th+\pi/2)*(z-z_a))$ determines the set of all points below of such straight line. When $0 < th \leq \pi$, the animat's perceptual region R_p is the semicircle defined by the intersection $((z-z_a)^2+(x-x_a)^2<r_p^2)\cap(x>x_a+\tan((th+\pi/2)*(z-z_a)))$; while when $\pi < th \leq 2\pi$, R_p is defined by the intersection $((z-z_a)^2+(x-x_a)^2<r_p^2)\cap(x<x_a+\tan((th+\pi/2)*(z-z_a)))$. Once determined the stimuli, which fall inside the animat's perceptual region, those stimuli that are behind an obstacle, are eliminated. In addition, the radius of perception r_p is proportional to the lucidity of the animat. The animat's perceptual region can be appreciated in Figure 4.

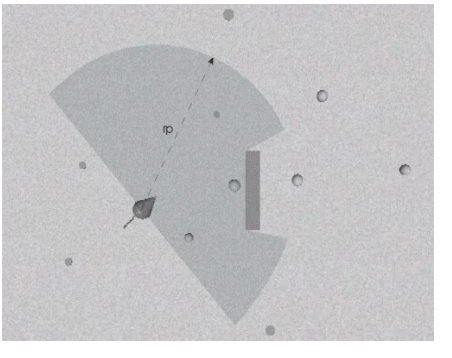


Fig. 4 Animat's perceptual region.

The perceptual system captures the magnitude of the stimuli and the distance between them and the animat. Then, the perceptual system sends to the ASM a pondered value corresponding to every kind of external stimuli, which is a function of the ratio between magnitude and distance for all the stimuli of the same kind. The pondered value represents the strength with which the kind of stimuli will be registered by the exteroceptors in the external perceptions blackboard level of the cognitive node.

Once the animat stops perceiving a stimulus (it is left behind the animat or an obstacle), the stimulus reverberates shortly in the perceptual system, decreasing its value, until it is very small and the system "forgets" the stimulus. This simulates, in the perceptual system, a short-medium time memory.

3.1.2 The Motor System

As can be seen in Figure 2, the walk of the animat is commanded by the angular steps α and β, with centre in the extremes of the diameter of the projection of the animat's sphere in the plane (z,x) perpendicular to the animat's orientation. The wander external behaviour is produced when angles α and β take random values between zero and one (radians), which describes a random trajectory. On the other hand, the exploration oriented to the search of a specific signal external behaviour is produced when angles α and β take equal values, which describes the animat's trajectory as a straight line. The size of the steps is proportional to the animat's strength.

3.1.3. The Internal Medium

The animat's internal medium is expressed through a set of variables, in which each one of them represents an internal state or need. For the actual simulation, the internal states strength, lucidity, security, fatigue, thirst and hunger have been considered. As it can be seen in Figure 3, the IBeNet is the only component of the animat's internal structure that establishes a direct interface with the internal medium. The IBeNet perceives each instant the actual value of each of the variables of the internal medium through the propioceptors; and updates such value in a determined proportion when the external behaviour associated to the internal state has been executed. For example, the execution of the consummatory behaviours eat and drink will reduce the values of the internal states hunger and thirst, respectively.

When the internal states hunger, thirst and/or fatigue are very high, the internal states strength and lucidity begin to decrease, slowing the movements of the animat and affecting his perception. On the other hand, if the internal states hunger, thirst and/or fatigue are satisfied, then strength and lucidity restore slowly. When strength reaches a value of zero, the animat dies.

3.2. The Simulated Environment

The animat's environment is defined by a plane (z,x) of a space (x,y,z). The plane (z,x) is delimited by a frame. In the area defined by this frame different objects can be created. This objects represent the external stimuli food (green spheres), water (blue circles), grass (texturized green circles), fixed obstacles (brown parallelepipeds), blobs (black ellipsoids), and other kinds of stimuli that initially have no specific meaning for the entity (red and yellow circles). The frame that delimits the plane (z,x) is also considered as a fixed obstacle. Figure 5 shows an air view of the simulated environment.

3.3. The Behaviour Repertory

The behaviour repertory (external actions) that the animat can execute is shown in Table 1. The selection of these behaviours responds to the properties of the IBeNet that are proved. Many of these behaviours can be executed only when both conditions, internal and external, have been satisfied. This is, there is a high internal need and the external

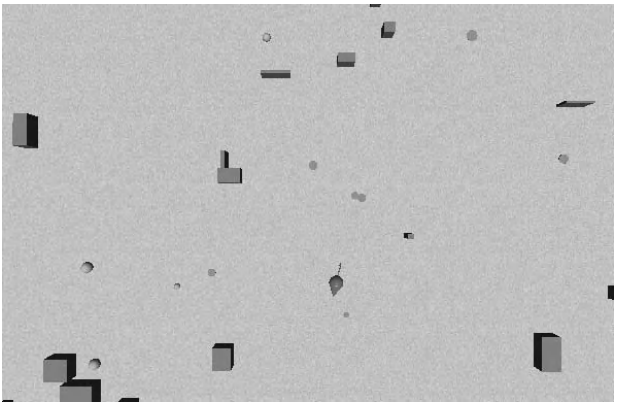


Fig. 5 Air view of the simulated environment

input capable of satisfying that need has been perceived. This is the case of the behaviours approach food, eat, approach water, drink, approach food and water, approach grass, sleep and runaway.

Table 1. Animat's behaviours repertory

Behaviour	External input	Internal input
Avoid obstacle	Obstacle at range	None
Wander	None	None
Explore	None	Thirst and/or hunger
Approach food	Food perceived	Hunger
Eat	Food at range	Hunger
Approach water	Water perceived	Thirst
Drink	Water at range	Thirst
Approach food and water	Food and water perceived	Hunger and thirst
Approach grass	Grass perceived	Fatigue
Rest	Grass at range	Fatigue
Runaway	Blob perceived	Safety

According to McFarland (McFarland and Houston, 1981; Maes, 1991), the associated behaviours with one or more motivations (or internal needs) are known as consummatory behaviours, while the ones that are not associated directly with some motivation are known as appetitive behaviours. In this way, a consummatory behaviour is that which an animat (or artificial entity) really wants to satisfy when the motivation associated to this is high; while an appetitive behaviour only contributes so that the consummatory behaviour can be executed.

In the animat simulation, we have considered that both kinds of behaviour exist: consummatory and appetitive. However, in difference to McFarland (McFarland and Houston, 1981), we have assumed that consummatory and appetitive behaviours are directly associated with some motivation. The difference between both kinds of

behaviours lies in that a consummatory behaviour is that which executes the final action required by the animat to satisfy a high internal need, and commonly reduces the strength of such need; while an appetitive behaviour contributes so that the consummatory behaviour can be executed, but the execution of the first does not reduce directly the motivation level to which it is associated. For example, if we consider that the approach water behaviour is appetitive and the drink behaviour is consummatory, then both behaviours will be associated to the thirst internal need; and the animat will only approach the water if it is thirsty, contributing in this way so that the desired final action, drink, will be executed.

4 Experiments

Let us consider an environment as the one described in section 3.2, in which there have been created objects O_i^+ that have a specific meaning for the animat, such as water, food, fixed obstacles, etc.; and objects O_i^n, which can acquire a new meaning for the animat. Let the positions in the environment, described as a plane, be defined by the pair (z, x). Let the entity animat defined by (1) a set of primitive actions that this can execute, as the ones described in section 3.1.2; (2) an ASM, as the internal behaviour network described in section 2; (3) a perceptual system, as the one described in section 3.1.1; and (4) a set of sensors of needs, that perceive the states of the internal medium.

The initial experiments were developed in order to verify: (1) the influence of the internal states in the observed external behaviour of the animat, (2) the role of the competition at a motivational level in the selection of the external behaviour to execute, (3) the exploratory behaviour oriented to the search of a specific signal and the reduction of the response times of the animat, (4) the stability in the selection and persistence in the execution of the external behaviours, (5) the discrimination between different stimuli taking in count the quality of them, (6) avoid aversive stimuli, (7) the non persistence in the execution of a consummatory action when an aversive stimulus is perceived, and (8) the role of the learning processes in the action selection. Next, experiments (2), (5), and (7) are presented and discussed.

4.1 The Role of the Competition at a Motivational Level in the Selection of the External Behaviour to Execute

Among all consummatory preferences selector elemental behaviours that have satisfied their condition, a competition of the winner takes all type takes place, in order to decide which of these behaviours will determine the external action that the animat will later execute. The behaviours that participate in this competition are incompatible behaviours, in the sense that only one will be able to send a drive signal directed to the cognitive node. A complete explanation of this competition is given in (González, 1999).

In order to illustrate this, a situation has been modelled, in which the animat has different internal states with significant and different values; at the same time that it is perceiving external signals capable to satisfy each one of these internal states. This modelled situation can be seen in Figures 6 and 7.

Figure 6 shows an initial state given by the position of the animat in the environment, the position and magnitude of external stimuli, and the initial values of the internal states thirst, hunger and fatigue. This is, the initial state defines the existent situation before the competition process at a motivational level takes place. Note that there is a need of drinking due to the very high value of thirst. As shown in Figure 6, the animat is perceiving plenty sources of food, water, and places where to rest.

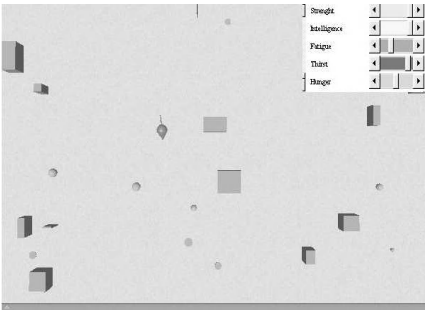


Fig. 6 Animat's initial state. There are thirst, hunger and fatigue; and there are sources of water, food and resting places.

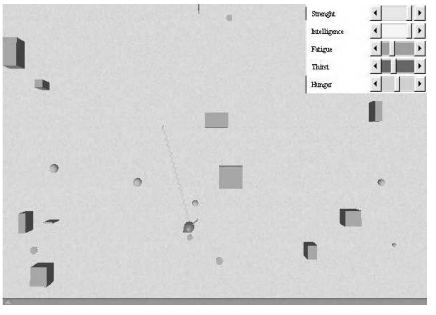


Fig. 7 State reached once the external behaviours approach water and drink were executed**.**

Since all of the animat's actual internal needs can be satisfied, because the corresponding external stimuli have been perceived, a competitive process takes place between the consummatory preferences selector elemental behaviours, in order to decide which behaviour must be executed. In Figure 7 a state is shown, in which the approach water appetitive behaviour was executed until the water source was at range, and the animat could execute the drink consummatory behaviour. In consequence of this, thirst was reduced.

Therefore, as it has been illustrated in Figures 6 and 7, every time that two or more external behaviours satisfy their conditions, a competition at a motivational level will take place in order to decide which is the most appropriate external behaviour for the animat to execute.

4.2 Discrimination Between Different Stimuli Taking in Count the Quality of Them

Let us have an initial state as the one shown in Figure 8, in which the animat has thirst and hunger, as much as thirst as hunger, and the animat perceives sources of water and food. In particular, it perceives an isolated source of water, an isolated source of food, and a source of food next to a source of water. Since these sources of water and food are next to each other, we consider both sources as a new kind of stimulus: source of water and food.

The "discrimination between different stimuli taking in count the quality of them" property establishes for the previous situation, that the best of this three stimuli is the source of water and food for the actual states of thirst and hunger of the animat. Once the

competition at a motivational level takes place, the first external behaviour that the animat executes is approach water and food.

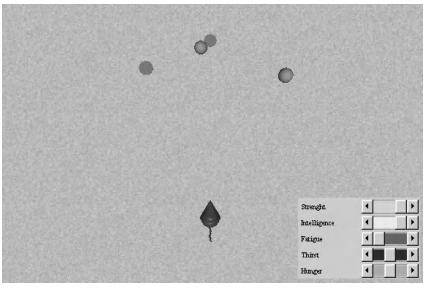


Fig. 8 Animat's initial state. There is thirst and hunger, as much thirst as hunger; and the animat perceives a source of food, a source of water, and a source of water and food.

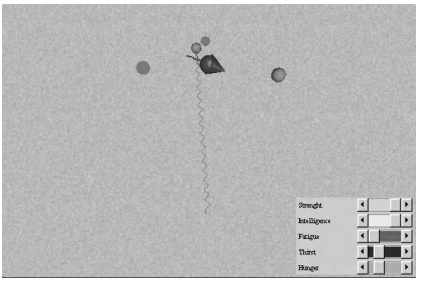


Fig. 9 State reached once the behaviours approach food and water, drink, and eat have been executed.

In Figure 9 is shown a state where the approach food and water external behaviour was executed until the source of water and food was at range for the animat. Then, the drink behaviour was executed, since the consummatory preferences selector elementary behaviour associated to the thirst internal state won the competition at a motivational level. As a result of the persistence of the drink consummatory action, the strength of the thirst internal state was decreased, which allowed the consummatory preferences selector elemental behaviour associated to the hunger internal state to win the competition at a motivational level, and as a result, the eat external behaviour was executed, until the strength of the hunger internal state would decrease enough.

The direct consequence that the animat had considered the source of water and food stimulus as the one with the best quality among all the stimuli perceived at that moment, was a reduction in the amount of external actions that the animat would need to execute in order to satisfy both his internal needs thirst and hunger. This is, once the thirst internal state was satisfied, the animat did not have to execute the approach food external behaviour in order to execute the eat behaviour, and satisfy the hunger internal state; since the food was at range and the eat behaviour could be executed without the precedence of an appetitive external behaviour.

4.3 The Non-persistence in the Execution of a Consummatory Action when an Aversive Stimulus is Perceived

The persistence in the execution of the external behaviours means that once the execution of the external behaviour has been initiated, the animat will always try to finish this, avoiding that other stimuli, irrelevant for the moment, distract his attention. Nevertheless, the non persistence in the execution of a consummatory action will take place when an aversive stimulus (blob) has been perceived and the ratio between the magnitude of the

blob and its distance to the animat will constitute a risk proportional to the given degree of the safety internal state of the animat.

In order to exemplify this property, let us consider an initial state as the one shown in Figure 10, in which the animat has a high degree of thirst and it is perceiving a water source, but has not perceived an aversive stimulus that is also in the environment.

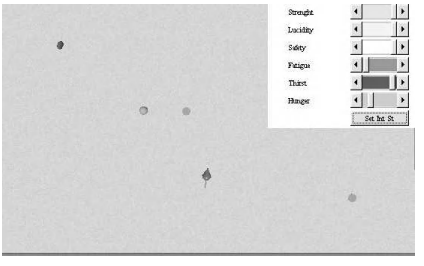


Fig. 10 Animat's initial state. There is a high thirst and water perceived.

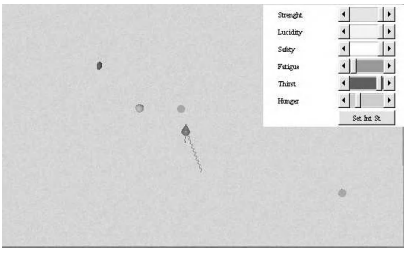


Fig. 11 The approaching water behaviour is being executed. The animat has not perceived the blob yet.

In Figure 11, a state is shown where the approach water behaviour is being executed, so that the animat is closer to the water source, but it still has not perceived the blob, which has moved and it is also nearer the water source.

As it can be seen in Figure 12, the animat continued the execution of the approach water behaviour, and once that this was at range, it began to execute the drink behaviour, which had to be interrupted before it fully satisfied the thirst need, due to the proximity of the blob. Therefore, the external runaway behaviour was executed. In Figure 12, it can be seen that the internal state thirst was decreased because of the execution of the drink behaviour, but it was not persistent enough to satisfy the need completely.

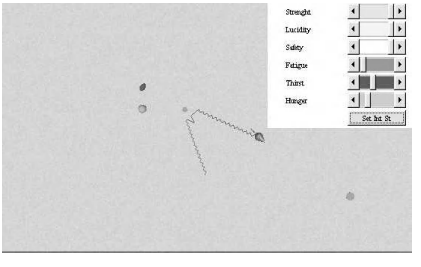


Fig. 12 The action drink is interrupted due to the proximity of the blob. The runaway action is executed.

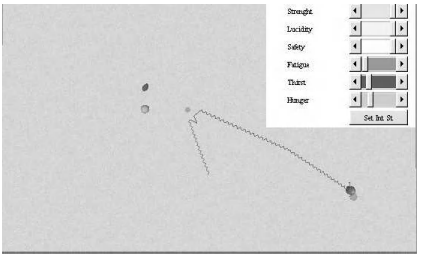


Fig. 13 Behaviours pattern that shows all the external actions executed by the animat.

As it is shown in Figure 13, once far of the blob, the animat begins to explore again in search of a water source in order to complete the consummation of the drink behaviour. A water source is perceived, the animat approaches it, and completes the interrupted action due to the presence of the aversive stimulus.

5 Conclusions and Future Work

The ASM that implements the IBeNet is characterized by the following properties, some of which could be appreciated in the experiments presented here, and the rest were verified in previous work (González, 1999):

- The external behaviour is influenced by the entity's internal states.
- Motivationally incompatible behaviours compete between them and the competition is at a motivational level.
- Modulation of reactive behaviour.
- Goal-oriented behaviour.
- Non indecision in the action selection.
- Stability in the selection and persistence in the execution of the external behaviours.
- Regulated spontaneity.
- Satiation.
- Changes in responsiveness.
- Varying attention.
- Preactivation of internal behaviours.
- Discrimination between the different kinds of external stimuli taking in count the quality of them.
- The existence of an external default behaviour oriented to the search of a specific signal accomplishes smaller response times, in respect of the satisfaction of an imperious internal need.
- Associative learning (classical primary and secondary conditionings).
- Reflex Response Inhibition.

While the simulation explained here was able to verify the main properties that characterize the action selection in the IBeNet, the implementation of the mechanism constitutes itself an ideal scenario for the modelling and testing of new behaviours and desired properties in the action selection of an autonomous agent, proceeding from areas such as reactive robotics, ethology, and cognitive sciences.

An immediate application of the IBeNet will be in the biomedical sciences area, in the modelling, comprehension, and prediction of biological system. Specifically, in protein interactions.

Among the future works there is the modelling of an artificial society of agents, where each agent will implement the IBeNet in a similar way as the one proposed here; and the integration of a previously proposed model if emotions (Gershenson, 1999) to the IBeNet, which will enhance the action selection.

6 References

Baerends, G. (1976) The functional organization of behaviour. *Animal Behaviour*, 24, 726-735.

Beer, R. (1990) Intelligence as Adaptive Behaviour: an Experiment in Computational Neuroethology. Academic Press.

Beer, R., Chiel, H. and Sterling, L. (1990) A biological perspective on autonomous agent design. *Robotics and Autonomous Systems*, 6, 169-186.

Brooks, R. A. (1986) A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation.* RA-2, April, pp. 14-23.

Brooks, R. A. (1989) A robot that walks: Emergent behaviour from a carefully evolved network. *Neural Computation*, 1, 253-262.

Gershenson, C. (1999) Modelling Emotions with Multidimensional Logic. *Proceedings of the 18th International Conference of the North American Fuzzy Information Processing Society* (NAFIPS '99), pp. 42-46. New York City, NY.

Goetz, P. and D. Walters (1997) The dynamics of recurrent behaviour networks. *Adaptive Behaviour*, 6(2), 245-282.

González, P.P. (1999) Redes de Conductas Internas como Nodos-Pizarrón: Selección de Acciones y Aprendizaje en un Robot Reactivo. *Tesis Doctoral*, Instituto de Investigaciones Biomédicas/UNAM, México.

González, P.P. and J. Negrete (1997) REDSIEX: A cooperative network of expert systems with blackboard architectures. *Expert Systems*, 14(4), 180-189.

Hallam, B.E, J. Halperin and J. Hallam (1994) An Ethological Model for Implementation in Mobile Robots. *Adaptive Behaviour*, 3 (1), pp 51-79.

Lorenz, K. (1950) The comparative method in studying innate behaviour patterns. Symposia of the Society for Experimental Biology, 4, 221-268.

Lorenz, K. (1981) Foundations of Ethology. Springer-Verlag.

Maes, P. (1990) Situated agents can have goals. *Journal of Robotics and Autonomous Systems*, 6(1&2).

Maes, P. (1991) A bottom-up mechanism for behaviour selection in an artificial creature. In J.A. Meyer and S.W. Wilson (ed.), *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behaviour* MIT Press/Bradford Books.

Maes, P. (1994) Modelling Adaptive Autonomous Agents. *Journal of Artificial Life*, 1 (1,2), MIT Press.

McFarland, D.J. and A.I. Houston (1981) Quantitative Ethology: The state space approach. Boston: Pitman.

Negrete, J. and M. Martínez (1996) Robotic Simulation in Ethology. *Proceedings of the IASTED International Conference: Robotics and Manufacturing*, Honolulu, Hawaii, USA, pp. 271-274.

Negrete, J. and P.P. González (1998) Net of multi-agent expert systems with emergent control. *Expert Systems with Applications*, 14(1) 109-116.

Nii, H. P. (1989) Blackboard systems. En A. Barr, P. R. Cohen, y E. A. Feigenbaum (ed.), The Handbook of Artificial Intelligence, volume IV, Addison-Wesley Publishing Company

Pfeifer, R. and C. Scheier (1999) Understanding Intelligence. MIT Press.

Rosenblatt, K. and D. Payton (1989) A fine-grained alternative to the subsumption architecture for mobile robot control. *Proceedings of the IEEE/INNS International Joint Conference on Neural Networks,* IEEE.

Tinbergen, N. (1950) The hierarchical organization of mechanisms underlying instinctive behaviour. *Experimental Biology*, 4, 305-312.

Tinbergen, N. (1951) The Study of Instinct. Claredon Press.

MultiAgent Planning: A Resource Based Approach

José Juan Palacios Pérez

Faculty of Computer Sciences,
B. Universidad Autónoma de Puebla, México
jpalacio@labpar.cs.buap.mx

Abstract. Interaction among agents is the distinguish property of MultiAgent Systems (MAS). In order to take advantage of such interaction w.r.t goals (both local for each agent and common for a group of agents) distributed planning, (i.e. the generation of coordination and cooperation of activities) is fundamental. Distributed MAS planning can be uniformly modeled in terms of resources: formal plan representation and effective interacting methods. Respectively, plan representation can be expressed by means of logical proofs and interacting methods by means of asynchronous communication. This paper presents preliminary considerations for the definition of a model for distributed MAS planning based on resources. For this mean, HACL (*Higher Order Asynchronous Communications in Linear Logic*) is used for analysis on the structural model RPN (*Recursive Petri Nets*) for MultiAgent distributed planning and for specification of MAS interaction protocols.

Keywords: Distributed Multi-Agent Planning, Interaction Protocols, Higher-Order Concurrent Linear Logic, Recursive Petri Nets, Semantic Model.

1 Introduction

Undoubtedly, interaction among agents is the distinguish property of MAS. In order to take the best advantage of interaction activities w.r.t. agent's goals, procedures for coordination and cooperation among agents are fundamental. This is achieved by means of distributed planning. Distributed planning involves two very strong aspects: there is no a global plan and the planning process must be dynamic in order to interleave plan construction and execution of actions. A very important problem to solve in distributed planning is to define efficient models of actions and plans, those models must take into account both concurrency of agent's activities and concurrency between agents and the enviroment.

The availability of resources is crucial in performing and modeling any kind of interaction: the more goods you have, the more activities you can perform. This consideration naturally leads to an uniform approach in which (in general) everything is a *kind of resource* (agents, actions, plans, enviroment), and interaction among those resources can be uniformly expressed by means of *asynchronous message passing* (i.e. production/consumption of messages).

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 649–660, 2000.

Linear Logic[Gir87] captures elegantly all these considerations: deductive inference in this formalism is a kind of concurrent resource management. This report has two main objectives: the first one is to show that Girard's Linear Logic is a very convenient formalism for reasoning about distributed planning in MAS. For this goal, the language HACL [KY94] is used for expressing and reasoning on *Recursive Petri Nets RPN* [SH96] which is a structural model for distributed MAS planning. The second and more ambitious objective is to establish a clear semantics that shows the close relationship between logic proofs and concurrent agent's actions. A natural consequence of this approach is to use it to formalize *communication protocols* for interaction in MAS, because a protocol is a type of plan in which actions are speech acts.

The plan of the paper is as follows: the important concepts and definitions of this approach are reviewed in the next section. Also, it outlines the RPN model for MAS planning and its interpretation in terms of HACL, establishing a correspondence between reachability in RPN with a proof in HACL. An example is used to illustrate the advantage of this approach. Dependency among actions, plans and agents is defined in the third section, as consequence of the kinds of interaction between them.

2 Logical and Structural Modeling

In this section we will present our conceptualization about the correspondence between a logical approach (based on Linear Logic) and a structural modeling (based on Recursive Petri Nets). In particular, the use of Linear Logic will be justified in terms of its expresivity for the notions involved. Briefly, Linear Logic[Gir87] is a dynamic non-monotonic logic which considers *formulae as resources* to be *consumed* or *produced* through application of *inference rules*. Everything is a kind of resource in this logic, so *deductive inference* can be considered as a form of concurrent resource management. We'll show the meaning of the connectives of this language through a description of interaction elements w.r.t. Multi-Agent systems (MAS).

2.1 Elements of Interaction

For our discussion, an *Enviroment* ($\mathcal{E}$) is a multiset of resources: consumable (also called *linear*) and reusable (prefixed with a *modality '!'*[1]). Indeed, each resource can be denoted by a propositional formula. A special kind of resources is given by the set of *Agents* ($\mathcal{A}gt$), which are autonomous dynamic resources, different from the others (more or less passive) resources. Let us denote $\mathcal{E} = \mathcal{R}es \cup \mathcal{A}gt$, where $\mathcal{R}es = \mathcal{C}Res \cup !\mathcal{R}es$, which denote consumable (*linear*) resources and reusable (*classical*[2]) resources, respectively. We'll consider two general types of resources: Agents and messages (messages are used to encode events and everything else).

[1] We will define this modality later.

[2] Each resource in $!\mathcal{R}es$ is prefixed by the modality '!' which indicates that the resource prefixed by ! can be used several times.

An *Agent* is a resource capable (by itself) of accomplish a given *task* by means of *actions* (resource consumption, production, etc.) on another resources (inclusive another Agents). Agents have two mechanisms in order to *interact*: an *interface* to perceive the enviroment and *effectors* to modify enviroment accordingly defined *goals* and/or *opportunistic choices* (in our approach, these mechanisms are uniformly captured by asynchronous message passing). An Agent is completely characterized by means of *attributes* and *behavior*. An *Identificator* (name, address) is a special attribute for denoting Agent's identity. The Agent's *state* is given by a *configuration* or collection of both available and reachable resources for the Agent, (in particular, reachable through *aquaintances*) and the *behavior* of the Agent is expressed by means of *Actions*. These actions can be modularly composed and abstracted, as we'll show. Without loss of generality, Agent behavior is *communication driven*, i.e. the incoming of messages (*events*) trigger the actions. Actions are expressed by means of transitions, which denote the basic interaction flow in terms of basic production - consumption of resources, i.e. establishing the *change of state*: $s \overset{Id.a}{\multimap} s'$ denotes the transition (labeled a) from *actual* state s to another state s'. A transition involves the identity (Id) of the agent who will perform the action. Note the connective used for denoting transition: '$\multimap$' is called *linear implication*. An expression $A \multimap B$ means that we can *deduce* (obtain) B by *consuming* the resource A.

The availability of resources in the enviroment is crucial, depending on them, the Agent should take the most adequate action, involving *non-determinism*. We can express two kinds of non-determinism in Linear Logic:

"inner" (denoted by '&') e.g. $A \overset{\alpha}{\multimap} B \& C$ which means the Agent in state A by means of action α can choose any one from A and B, *but not both at the same time*;

"outer" (denoted by '$\oplus$') in which the election depends of the availability of the resources involved in that instant of time, e.g. $A \overset{\alpha'}{\multimap} D \oplus C$.

Concurrency can be expressed naturally by means of connectives of Linear Logic: the *multiplicative conjunction* $\otimes$ in $A \otimes A \multimap B \otimes C$ means that both (two instances) resources A must be available at the same time in order to produce B and C together. '$\parr$' denotes the *asynchronous* counterpart of $\otimes$ (these will be used for expressing communication, as we'll show below). The most important notion in Linear Logic is *duality*, this is denoted by means of the *linear negation* '$.^{\perp}$', it allow us to express *input/output, question/answer, action/reaction*[3], etc.

A number of considerations are neccesary in order to allow an action to be executed: let us denote $\mathcal{C}Res_a$ as the multiset of consumable resources necessary

[3] Intuitively, our notion of deduction can be expressed as :

$$\textit{action of type } A \vdash \textit{reaction of type } A^{\perp}$$

Remarkably, Linear Logic refines the concept of *theory* (traditionally *classical logic + axioms*), to [Gir95]:

$$\textit{theory} = \textit{Linear Logic} + \textit{Axioms} + \textbf{current state}$$

in order to apply /allow action a (i.e. a function $Pre(a) : r \mapsto r'$, r, r' multisets contained in $\mathcal{R}es$), $\mathcal{P}Res_a$ as the multiset of new resources configuration after application of a: $Post(a) : r \mapsto r'$. So, the *conditions* that must be hold for an action be applied are: $s' = s \backslash \mathcal{C}Res_a \cup \mathcal{P}Res_a$; $\mathcal{C}Res_a = Pre(a)$ (except when there exists some non-consumable resource $!p$) and $\mathcal{P}Res_a = Post(a)$.

Actions can be expressed uniformly by means of *message passing*: by using predicates in our Linear Logic formulae, an atomic formula $m(a)$ can be interpreted as a message carrying a value of a, so a predicate m can now be considered as a *communication channel* or a port name. Then, the formula $\forall x.(m(x) \multimap A(x))$ means that for any x, if $m(x)$ holds, we can deduce $A(x)$, i.e. "for any x, if there is a message m carrying x, we obtain a *process expression*[4] $A(x)$". Therefore, this formula represents a process which receives a *value* of x via a message m and becomes $A(x)$:

$$m(a) \otimes \forall x.(m(x) \multimap A(x)) \otimes \Gamma \multimap A(a) \otimes \Gamma$$

(Indeed, Γ denotes the *context*, i.e. configuration of resources in the system (Agents - process expressions, multisets of messages, etc.)) In general we differentiate two kinds of actions: *elementary* (basic one, irreducible or atomic) and *abstract* (composite action that can be performed in parallel -independent of others- or exclusive -requires coordination with another actions). Execution of abstract actions involve the *refinement* of behavior. This refinement is realized by means of *method invocation*, where the methods are expressed by *clauses*[5] of the form: $\forall A_p \circ\!\!- G$ whose meaning is "a behavior expression (denoted by a *process predicate* can become the new behavior denoted by G", G is called a *goal formulae*. Actually, given all these elements, we can *denote* the evolution of a configuration of Agents and messages by means of a *formal proof*, in which transitions are identified with the inference rules associated with the connectives. This notion (*proof search as computation*) is summarized in the following table:

Linear Logic		Agent Expression
formula	$\rightarrow$	process (behavior), message (event)
sequent	$\rightarrow$	configuration (enviroment)
proof	$\rightarrow$	trace of computation
inference rule	$\rightarrow$	transition (actions)

This correspondence leads naturally to consider those proof structures (proof trees) as *Agent's Plans*, i.e. a set of actions under a certain order relation, as we'll show below.

Briefly, this is precisely the syntax and intuitive semantics of the language *HACL: Higher-Order Asynchronous Communication in Linear Logic* [KY93], [KY94] which we resume in the table below:

The axioms are forever (using formulae !'ed), but the current state is available for a *single use:* hence once it has been used to prove one state, the theory is *updated* and this state becomes the next current state.

[4] Process expressions denote behavior of an Agent.

[5] Exactly as in Logic Programming.

Formula	Description
$Cl ::= \forall \boldsymbol{x}(A_p(\boldsymbol{x}) \circ\!\!- G)$	Clause
$A_p(\boldsymbol{x})$	Process predicate
$G ::= \bot \mid \top \mid A_m(\boldsymbol{e}) \mid ?A_m(\boldsymbol{e}) \mid A_p(\boldsymbol{x}) \mid$ $G \parr G \mid G \& G \mid R \mid \forall \boldsymbol{x}.G$	Goal
$A_m(\boldsymbol{e})$	Message predicate
$R ::= \exists \boldsymbol{x}(M \otimes G) \mid R \oplus R$	Message reception
$M ::= A_m^{\perp}(\boldsymbol{x}) \mid M \otimes M$	multiset of messages

$\bot$ indicates process'suicide, $\top$ is the normal finalization, $\parr$ indicates parallel construction and asynchronous message send.

This language enjoys a rich expresivity: due to the High-order, both process expressions and messages are *first class objects*, i.e. they can be send by messages. This property allows to construct *dynamic communication topologies*, defining Agent's structure by incremental composition (hierarchical construction independent of a particular implementation of components) enhacing modularity. Aditionally, HACL offers a ML-style static polymorphic *Type System* in which the most important notion is the identification of a well typed process as a well typed formulae. Accordingly, recursive typing, polymorphic records, and *Object Oriented style of programming* can be obtained by enriching the basic type system.

Therefore, interaction (both internally and with the external world) is expressed by means of communication[6]. As a consequence, coordination facilities (planning management) in the Agent's community is also uniformly expressed in terms of communication.

Following [Whi96] we can justify this proof-theoretic approach for denoting *plans as proofs*:

- Actions are the referents of Linear Logic proofs, (establishing a sort of connection between Linear Logic and reality) because proofs are *intensional* and actions are intensional too.
- Actions describe, both the state transition and how it was brought about; this is the sort of intensionality that linear proof theory offers.
- Actions only change-consume those resources whose the actions are applied, the effect of an action for the remain resources is "no change": this property clearly avoids the frame problem.
- Basic or primitive actions are expressed in terms of the rules associated with the logical connectives.

Continuing with our notions, we consider (as usually) a *Plan* ($\mathcal{P}l$) as a finite set of actions $\mathcal{P}l = \Theta(a_i)_{i \in I}$, sorted under a *partial order* (allowing that two

[6] Ok, not all interaction is plain message passing, but, without loss of generality, we can consider uniform interfaces in terms of sending and receiving messages, because the higher-order allow us to encode (almost) any kind of resource that can be perceived in the enviroment, i.e. messages are abstract (and expressive) enough to encode virtually any perceivable event.

actions can be concurrent). A plan is said to be *feasible* if and only if whatever the total order that extends the partial order, every action $a_i \in \mathcal{P}l$ remains allowed w.r.t. the order. We briefly describe here the structural model for denoting plans: *RPN: Recursive Petri Nets* [SH96]. In essence, this model denotes a Plans by means of a net, in which *markings* (tokens in places) denote the state configuration and the firing of transitions denote the execution of actions. In this model, actions are performed by *Methods* [7]. A method consists of:

- An *identifier*,
- an abstract attribute that denotes its kind: elementary or abstract,
- a set of *initialized RPN* ($IRPN =< R, M_0, Bind >$) (in the case of an abstract method) where:
  - R is a RPN (skeleton)
  - $M_0 : P \mapsto N$ is the *initial marking* of R, (initial configuration, safe state); P is the set of places in a RPN (defined below) and N is the set of natural numbers.
  - $Bind : Var \mapsto D$, a total or partial link from variables of R to the *domain objects* (D), allowing to instantiate also the parameters of the methods.

There are two kinds of methods: elementary and abstract. The elementary ones just call the subroutine associated with them, while the abstract ones refine the plan in terms of sub-plans according to the chosen refinement.

An RPNet $R =< P, T, Pre, Post, Call, Var >$ consists of the following elements:

- A set of Places P,
- a set of Transitions T, (arcs between places: elementary, abstract and final)
- Preconditions: $Pre : P \times T \mapsto N$ denoting the places that may be *marked* (tokens) in order to activate (fire) a transition (action) t;
- PostConditions: $Post : P \times T \mapsto N$, indicate the tokens available after a transition firing t;
- a set of Variables Var,
- Methods Invocations: $Call(t)$ denotes the call of a method associated with the transition t, involving the following components:
  - identifier of the method,
  - an expression (involving variables in Var)which denotes the agent who calls the method,
  - an expression of the call parameters for the method (*i.e. an activation frame)* denoting the Pre and Post conditions of the associated action t, (hence the recursion involved).

Therefore, a plan execution is represented by the RPN model according to the previous definitions:

[7] Indeed, they are equivalent with the HACL methods (clauses) presented above.

- initial marking M_0 allows the plan execution to start,
- Pre - Post conditions satisfied: $Pre(p,t)$ (respectively $post(p',t')$) returns $n > 0$ if the place p is an input place (respectively, output place) of the transition t and the valuation of the arc linking p to t (respectively, t' to p') is n (i.e., ready to fire). Otherwise returns 0.

Example: We have a system composed of an agent *Manager* and a group of agents *Colleagues*. The Manager acts as a leader project, receiving requests for performing some task. Colleagues work in cooperation for solving the task and must share a kind of resources (*forks*) in order to do their job. They must coordinate themselves, and send their results back to the Manager. We can express the behavior specification for each agent by means of clauses, the left part denotes the multiset of requierments (messages) in order to execute the method. $lfork(id) \otimes rfork(id-1) \overset{do_job(rst)}{\multimap} rst(Myjob)$ this first specification (clause) means that the colleague (id) has to wait for the availability of the 2 forks, and when they are available, it can change its behavior in the goal denoted by $do_job(rst)$, where rst means a channel in where the finalized task (bunch) will be sent back.
$rst(Myjob) \overset{result}{\multimap} bkjob(Myjob,id) \parr lfork(id) \parr rfork(id-1)$ in this goal specification, the colleague waits for the message result (rst) that subprocess (abstract method) do_job will send. The manager has the following goal specification: $request(who, Job, Self) \overset{request()}{\multimap} schedule(acqnts, Job)$, in this case Job is a goal specification, i.e. requierments to be solved by the team. $Self$ denotes the address (identity) of the manager, the sender's identity is denoted by who [8].

Manager agent has an Actor-like structure: a *mail-address (self)*, a group of *aquaintances* (the colleages), operations for new agent's creation (auxiliar *schedule*), change behavior and send messages. Manager and colleagues work like a primitive organization. The structure for the group of colleagues is defined in terms of the shared channels (forks), indeed, any given topology can be established. The colleagues can be incorporated in the Manager as follows:

```
proc man-and-colle (id) =
 manager (ring( nlist 5 colleague)) id;
```

where *ring* creates a round-robin list of agents.

2.2 Correspondence HACL - RPN

We can express the correspondence from RPN to HACL as follows:

- Implicity, places denote an Agent behavior: a multiset of terms to be consumed, either in a communication (by agent initiative) or a process reduction (responsability of system organization). Transition places are denoted indeed by '$\multimap$'.

[8] Due to space limitations, the HACL code is not showed.

- Tokens in places on a net (*marking*) correspond to a multiset of processes and messages in the enviroment (situation), we can differentiate two basic kinds of tokens:
  - Agents, have as value their identificator (address)
  - Messages, involve addresses, binding variables, a certain structure and the communication itself.

  Both of them are created by the $\parr$ connective.
- Firing of a transition corresponds to the action execution involved: incoming arcs are $\otimes$'ed together (preconditions), outcoming arcs are $\parr$'ed. Multiple choice is denoted by & and $\oplus$ on the preconditions and postconditions respectively.
- Pre- conditions: messages (and process configuration) available which are consumed in order to change the agent's behavior. Post- conditions are new messages (new tokens in places), results of function application, etc. In other words, formulae that express the states of the world before and after an action achievement.

Evolution of a net (behavior) refers how markings change as transitions occur, i.e. reachability from a given marking. For markings m, m' and a transition $t \in T$ (T is the set of transitions in the net), $m \xrightarrow{t} m'$ means t fires from m to m' *iff* $\exists m'', {}^{\backprime}t, t' \in \mathcal{M}, m = m'' + {}^{\backprime}t$ and $m' = m'' + t'$ (where $\mathcal{M}$ is the set of all markings, ${}^{\backprime}t$, t' is respectively pre- and post- multisets associated with transition t). Therefore, we define the *reachability relation* $m \to m'$ as:

$$m \to m' \textit{ iff } \exists \boldsymbol{t_i} \in T \boldsymbol{m_i} \in \mathcal{M},$$
$$m \xrightarrow{t_1} m_1 \ldots \xrightarrow{t_n} m_n = m', (0 \leq i \leq n).$$

For the purposes of Linear Logic interpretation, we are interested into justify what the *sufficient requirements* for a given marking are (because these requirements are precisely captured by the stages of a proof): we define the *downwards closure* $\downarrow(m) = \{m' \in \mathcal{M} \ / m' \to m\}$. This is clearly an order relation. So, our interpretation of markings in terms of Linear Logic is as follows [Tan98]:

- Atomic propositions corresponds to markings on a net
- An assertion A is denoted by a set of markings $[\![A]\!]$, $m \in [\![A]\!]$ means that m is a sufficient distribution of resources to establish A according to the net.
- Constants: $[\![\top]\!] = \mathcal{M}$, a sufficient distribution for a sucessful termination is given by the whole set of configurations; $[\![F]\!] = \emptyset$ falsity denoted by the empty configuration, $[\![a]\!] = \{m/m \to a\}$ for a constant a, the associated configuration is indeed the downwards closure of a.
- Compound formulae:
  1. $[\![A \otimes B]\!] = \{m/\exists m_A \in [\![A]\!], m_B \in [\![B]\!], m \to m_A + m_B\}$, the multiset union of the configurations for both A and B
  2. $[\![A \multimap B]\!] = \{\ m/\exists m_A \in [\![A]\!], m + m_A \in [\![B]\!]\}$, this can be expressed as 'which are the requierements in addition of the ones for A that are neccesary to establish the requierements of B?'
  3. $[\![A \& B]\!] = [\![A]\!] \cap [\![B]\!]$,
  4. $[\![A \oplus B]\!] = [\![A]\!] \cup [\![B]\!]$

Plan	RPN	HACL
Current State	Place(s)	Agents configuration
Action	Transition	Proof
Method to execute (t)	Call(t)	Method (clause) invocation
Variables of Plan	Var(X.Id)	Variables, names and scoping
Pre- and Post conditions of action	Pre- and Post-conditions Matrix	Resource production/consumption

Fig. 1. Correspondence between RPN and HACL

In other words, there exists a correspondence between a reachable place (firable transition) in a net and the probability of the associated formula in Linear Logic:

Proposition 1 *The sequent $A \vdash A'$ is deducible in Linear Logic if for any marking m in the interpretation $[\![A]\!]$, there exists $m' \in [\![A']\!]$ such that m is reachable to m': $A \vdash A'$ if $[\![A]\!] \subseteq [\![A']\!]$.*

Therefore, the interpretation $[\![A]\!]$ can be justified by a proof in HACL:

Schedule of the Proof: induction on the length (abstraction, i.e. refinement) of the transition(s) involved, w.r.t. the logical connectives involved in the HACL formulae. Show it by cases on the operators (actions) involved, three kinds: basic (funcion application), communication and activation of methods (refinement of actions).

The correspondence between RPN and HACL (as far as distributed planning is concerned) is sumarized in the figure 1.

3 Conditions of Interaction

In this section, we'll show the usefulness of our approach as far as multi-agent plan management is concerned. Fist we characterize the kinds of interaction in a bottom-up fashion: from characterizing dependences between actions, plans up to Agents.

We are interested in dealing with multi-plans (both for one agent and several agents), so we can consider in general two kinds of interaction: *positive* (+) and *negative* (−). So, we first define these kinds of interactions w.r.t resources in a bottom-up direction:

3.1 Interactions between Actions

Formally, we define 2 cases both for sequential and concurrent actions:

1. sequences of actions (bottom -up definition):
   - (+)int: if $\mathcal{C}Res(a_2) \cap \mathcal{P}Res(a_1) \neq \emptyset$ then $a_1 Ip^* a_2$, where (Ip^* denotes a sequence of positive interactions relation -i.e. transitive closure)
   - (−)int: if $\mathcal{C}Res(a_1) \cap \mathcal{C}Res(a_2) \neq \emptyset$ then $a_1 Ina_2.In$ means *negative interaction*.
2. concurrent actions:
   - (+)int: $\emptyset$ (we cannot reason about the $\mathcal{P}Res$)
   - (+)int: general case from above $\mathcal{C}Res(a_1) \cup \mathcal{C}Res(a_2) \neq \emptyset$

3.2 Interactions between Plans

We can establish more generally:

1. if exists $a_1 \in Pl_1$, such that $\{a_1\}Ip\{a_2\}$ means that $Pl_1 IpPl_2, a_2 \in Pl_2$, Ip means positive interaction relation; in the same idea, for negative interactions if $\{a_1\}In\{a_2\}$ means $Pl_1 InPl_2$
2. if exists $a_1 \in Pl_1$; exists $a_2 \in Pl_2$ such that $\mathcal{C}Res(a_1) = \mathcal{C}Res(a_2)$, $\mathcal{P}Res(a_1) = \mathcal{P}Res(a_2)$ and for all $r \in \mathcal{P}Res(a_i), (i = 1, 2)$ r is not consumable (i.e. it is a reusable resource $?r$).

3.3 Interactions between Agents

As consequence, the interaction between agents can be expressed as follows: if $\exists a_1 \in Pl_1$ such that $id(a_1) \neq id(Pl_1), id(Pl_1) \in Agt$ then $id(Pl_1)$ *"depends of"* $id(a_1)$. (the relation "depends of" is not a transitive relation.)

Given those definitions, we can solve these kinds of interaction by means of the following general strategies:

- (+) int: delegation of the (redundant) task (structural solutions), conditions in terms of Agent's Id.
- (−)int: we have two cases
  1. resource conflict (consumable resources / reusable/ ...)
  2. concerning some effects of Preconditions, use of heuristics for reducing the interval (synchronization).

As illustration, let us suppose that a couple of Agents colleagues from our previous example need to coordinate because one of them does not know how to solve a task (in the third clause, $X.t_3$, indicates that the variable X is not instantiated with the Agent who will perform it), he just know what are the preconditions and what the action should return. So, he ask to the other agents if one of them can execute that action for him. Let be the following Agent's plans:

Agent_1:	Agent_2:
$q_o \xrightarrow{Self.t_1()} q_2 \parr q_3 \parr \bot$	$p_o \xrightarrow{Self.r1()} p_1 \parr \bot$
$q_2 \xrightarrow{Self.t_2()} q_4 \parr \bot$	$p1 \xrightarrow{\mathbf{Self.t3}()} p2 \parr \bot$
$q_3 \xrightarrow{\mathbf{X.t_3}()} q_5 \parr \bot$	$p_2 \xrightarrow{Self.end()} \bot$
$q_4 \otimes q_5 \xrightarrow{Self.end()} \bot$	

So, after merginig plans:, Agent_1 instantiates the var. $X == Agent_2$, and $Agent_2$ sends the result back to Agent_1, refining the method $t3()$ for the corresponding action: Agent_1's action t_3:

$$p_1 \xrightarrow{Agent_2.t_3()} q_5 \parr \bot$$

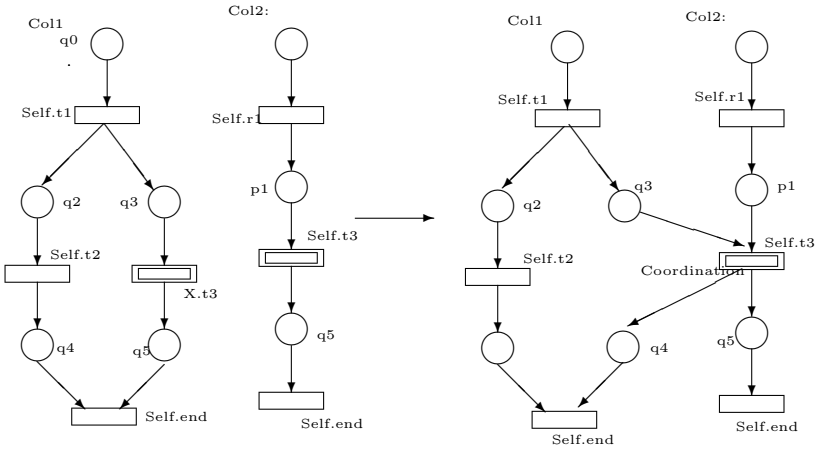


Fig. 2. Coordination between 2 agents

Agent$_2$'s action t_3:

$$p_1 \otimes q_3 \overset{Self.t_3()}{\multimap} p_2 \parr q_5 \parr \bot$$

Reachability conditions still hold:

$$\begin{array}{ll} [\![q_2]\!] = \{q_2, q_0\} & [\![q_5]\!] = \{q_3, q_5, q_0\} \\ [\![q_4]\!] = \{q_4, q_2, q_0\} & [\![q_5 \oplus q_2]\!] = \{q_3, q_2, q_5, q_0\} \\ [\![q_5 \otimes q_4]\!] = \{\{q_o, q_o\}, \{q_2, q_3\}, \{q_4, q_5\}\} & [\![q_2 \multimap q_5]\!] = \{q_3\} \end{array}$$

$$\models [\![q_0]\!] \multimap [\![q_5 \otimes q_4]\!]$$

4 Conclusion and Future Work

HACL satisfies the MultiAgent distributed planning requirements [SH96]:

- Adequate plan representation: it is given in a consistent and uniform way (supported by the higher-order) by means of proof search in HACL, offering multiple abstraction levels.
- Efficient interacting methods: the interaction is communication driven (in all levels of abstraction: in the compositional (intra-agents) process, inter-agents, and organizations).

HACL is an adequate formalism for reasoning on RPN plans: is a rich, simple and robust language.

As work in progress, we are currently designing a visualization tool: given an agent's organization description (perceptions, actions, goals, plan and enviroment, relationships between agents, policies and structure of organization, time constraints and location dependences) encoded in (extended) HACL, to get a number of propierties and policies of plan management (i.e. identification of redundant actions, merging of plans, etc.) graphically in terms of RPNs.

Acknowledgments. My sincere thanks to Amal El Fallah Seghrouchni and to the anonymous referees for their comments.

References

Gir87. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987. The seminal paper of Linear Logic.

Gir95. Jean-Yves Girard. Linear logic: Its syntax and semantics. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 1–42. Cambridge University Press, 1995. Proceedings of the Workshop on Linear Logic, Ithaca, New York, June 1993.

KSY98. Naoki Kobayashi, Toshihiro Shimizu, and Akinori Yonezawa. Distributed concurrent linear logic programming. Technical Report 94-12, Dpt. of Information Science, University of Tokyo, 1998.

KY93. Naoki Kobayashi and Akinori Yonezawa. ACL — a concurrent linear logic programming paradigm. In D. Miller, editor, *Proceedings of the 1993 International Logic Programming Symposium*, pages 279–294, Vancouver, Canada, October 1993. MIT Press.

KY94. Naoki Kobayashi and Akinori Yonezawa. Higher-order concurrent linear logic programming. In *Proceedings of Theory and Practice of Parallel Programming (TPPP'94), Sendai, Japan*, Lecture Notes in Computer Science (to appear), 1994.

SH96. Amal El-Fallah Seghrouchni and Serge Hadad. A Recursive Model for Distributed Planning. *ICMAS'96*, 1:7–38, 1996.

Tan98. Makoto Tanabe. Timed Petri Nets and Temporal Linear Logic. Technical Report 98-2, Research Institute for Mathematical Sciences, Kyoto University, Japan, 1998.

Whi96. G. Graham White. The design of a situation-based lygon metainterpreter: I. simple changes and persistence. Technical Report 729, Department of Computer Science, Queen Mary and Westfield College, University of London, 1996.

Dynamic Fuzzy Logic

José Luis Pérez-Silva and Felipe Lara-Rosano

Laboratory of Applied Cybernetics, Centre for Instrumentation Research, National University of Mexico (UNAM).. Apdo. 70-186 CP 04510 Coyoacan, Mexico D.F.
Tel: (525) 56228608 Fax: (525) 56228617
pepito@aleph.cinstru.unam.mx

Abstract. Departing from the notion of a dynamical fuzzy set, we extend the concept of fuzzy logic, to introduce the dynamic fuzzy logic. The dynamic fuzzy logic is the base of the dynamic approximate reasoning, where the truth values and the inference rules are fuzzy, and change over time. The meaning of a dynamical fuzzy conditional proposition of the form IF A(t) THEN B(t), is clarified. We define also a dynamic linguistic variable as a variable whose truth value is represented as a word or sentence in a natural or artificial language whose meaning changes over time.

1 Basic Concepts in Dynamic Fuzzy Logic

Departing from the theory of dynamical fuzzy sets, we can extend the notion of fuzzy logic to introduce the dynamic fuzzy logic. Let p (t) and q (t) two propositions with time varying truth values P(t) and Q(t) respectively in the closed interval {0,1}. The truth values of the basic connectives are defined as:

$$\left|P(t) \wedge Q(t)\right| \overset{t}{=} \min\left(P(t), Q(t)\right), \quad \left|P(t) \vee Q(t)\right| \overset{t}{=} \max\left(P(t), Q(t)\right),$$

$$\left|P(t) \rightarrow Q(t)\right| \overset{t}{=} 1 \text{ if } P(t) \leq Q(t), \quad \left|\neg P(t)\right| \overset{t}{=} 1 - \left|P(t)\right|,$$

In dynamic fuzzy logic can define the dynamic fuzzy link as following:

$$P(t) \left| \overset{T}{\rightarrow} Q(t) \text{ if } P(t) \leq Q(t) \right.$$

For instance: Let p (t) = Juan is very high in the time t and q (t) = Juan is high in the time t. Clearly p(t) links semantically to q(t), since at the time t the truth value of "very high" could be smaller or equal to the truth value of "high", but at another time this has not to be true.

"$\rightarrow$" is the logical correlative of material implication. In dynamic fuzzy logic we have following tautologies for all time:

$$P(t) \overset{T}{\rightarrow} P(t) \forall t \text{ and } \neg P(t) \overset{T}{\wedge} P(t)$$

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 661—670, 2000.

For instance, let us suppose that p(t) = *this car is red at time t.* and in fact at time t the car is 60% red and 40% other colors. Then we can say that P(t) = 0.6 . In the same way $P(t) \vee \neg P(t)$: "*The car is red or not at time t*", has a truth value of 0.6 in that time t, according to the given semantics. Similarly: $P(t) \wedge \neg P(t)$: *"The car is red and not red in the time t"* is not a contradiction, and its truth value is 0.4 in the time t. At another later time t' P(t') could be only 0.3 because the owner decided to paint some parts with another color or the sun provoked that the color changed. Then the new value for the proposition $P(t) \vee \neg P(t)$: "*The car is red or not at time t*", is 0.7

In terms of the dynamic fuzzy logic the Modus Ponens can be generalized of such form that it preserves the degree of truth. If we consider P(t) = a, and we know that $p(t) \rightarrow q(t)$, then q(t) is truth in degree a at time t, that is Q(t) = a

$$\left|\overset{T}{-}_{\alpha} p(t)\,,\ p(t) \overset{T}{\rightarrow} q(t)\,,\ \therefore \left|\overset{T}{-}_{\alpha} q(t)\right.\right.$$

Following classic tautologies are also valid in the propositional dynamic fuzzy logic:

$$P(t) \overset{T}{\rightarrow} P(t)$$

$$\left(P(t) \overset{T}{\rightarrow} \left(Q(t) \overset{T}{\rightarrow} R(t)\right)\right) \overset{T}{\rightarrow} \left(P(t) \overset{T}{\rightarrow} Q(t)\right) \overset{T}{\rightarrow} \left(P(t) \overset{T}{\rightarrow} R(t)\right)$$

$$\left(\neg P(t) \overset{T}{\rightarrow} \neg Q(t)\right) \overset{T}{\rightarrow} \left(Q(t) \overset{T}{\rightarrow} P(t)\right)$$

$$\neg\neg P(t) \overset{T}{\rightarrow} P(t)$$

$$\left\{\left(P(t) \overset{T}{\rightarrow} Q(t)\right) \overset{T}{\wedge} \neg Q(t))\right\} \overset{T}{\rightarrow} \neg P(t)$$

$$\left(P(t) \overset{T}{\rightarrow} Q(t)\right) \overset{T}{\rightarrow} \left(\left(Q(t) \overset{T}{\rightarrow} R(t)\right) \overset{T}{\rightarrow} \left(P(t) \overset{T}{\rightarrow} R(t)\right)\right)$$

Morgan laws.
Asociatives laws.
Distributive laws.
Conmutatives laws.

The not valid tautologies in the propositional dynamic fuzzy logic are:

$$P \overset{T}{\vee} \neg P \text{ for a t}$$

$$\left(P(t) \overset{T}{\rightarrow} \left(Q(t) \overset{T}{\wedge} \neg Q(t)\right)\right) \overset{T}{\rightarrow} \neg P(t) \text{ for a t}$$

$$\left(P(t) \overset{T}{\wedge} \neg P(t)\right) \overset{T}{\rightarrow} Q(t) \text{ for a t}$$

The dynamic fuzzy logic is reduced to the propositional fuzzy logic when the propositionals variables are valid for all time.

2 A Dynamic Fuzzy Predicate Logic

We can deduce by extension the dynamic fuzzy logic predicates of the propositional dynamic fuzzy logic, defining dynamical predicates and dynamical quantificators. Let F(t) a predicate of n values. We defined the value of $Fx(t)_1,...,Fx(t)_n$ as the membership degree of the ordered sequence $(x_1(t),...,x(t)_n)$ to the fuzzy set F (t), where $x_i(t)$ is the denotation of x(t) in a given valuation i. In other words:

$$\left|F(t)_{x_1},...,F(t)_{x_n}\right| = \psi(t)\left(\overline{x}_1(t),...,\overline{x}(t)_n\right)$$

The quantificators are valued of the following form: $\forall x(t)\ F(t)_x$ is defined as the minimum of the values of F(t)x(t) for all the assignments of x(t) to the elements of the universe of discourse.

The value of $\exists x(t)F(t)_x$ is the corresponding maximum, or rather:

$\left|\forall x(t)F(t)_x\right| \overset{T}{=} \min\left\{\left|F(t)_x\right|\right\}$ for all the assignments of x (t) in the universe of discourse

$\left|\exists x(t)F(t)_x\right| \overset{T}{=} \max\left\{\left|F(t)_x\right|\right\}$ for all the assignments of x (t) in the universe of discourse.

3 A Dynamic Fuzzy Modal Logic

We can have a modal dynamic fuzzy logic, adding the operators "$\overset{T}{\uparrow}$" as the necessity functor and "$\overset{T}{\downarrow}$" as the possibility functor, and giving them the values:

$$\left|\overset{T}{\uparrow}P(t)\right| / W(t) = \min\{\left|P(t)\right| / W(t)\} \text{ for all W (t) such that R(t)WW.}$$

$$\left|\overset{T}{\downarrow}P(t)\right| / W(t) = \max\{\left|P(t)\right| / W(t)\} \text{ for all W (t) such that R(t)WW.}$$

where"/" is the alternative relationship. In the modal dynamic fuzzy logic there are degrees of the need of truth. Therefore the value "$\overset{T}{\uparrow}P(t)$" is equal to some α, $0 < \alpha < 1$. It is interpreted as that p(t) is necessarily truth with a degree α.

4 A Dynamic Fuzzy Presuppositional Logic

Let us consider the presupposition as: p(t) presupposes q(t), if

$$p(t) \Vdash q(t) \text{ and } \neg p(t) \Vdash q(t)$$

If we wish to make an fuzzy extension keeping the evaluation:

$$\left(\left|\neg p(t)\right|\right) = 1 - \left|p(t)\right|$$

taking the link definition we have: p(t) presupposes q(t), if:

$$P(t) \leq Q(t) \text{ and } 1 - P(t) \leq Q(t) \text{ in all the models.}$$

But this means that P(t) presupposes Q(t) only if Q(t) = 0.5 at time t, that it is a strange result.

A form of solving this problem is instead of assigning only a truth value to the proposition, we assign an ordered couple: a truth value and a falsity value: (V(t), F(t)), whose sum is: V(t) + F(t) = 1 for all times. For the temporary presuppositional logic this can be extended by adding a third value N (t) to the valuation set (V(t), F(t), N(t)) so that:

V (t) + F (t) + N (t) = 1, where N (t) is what makes no sense. Thus:

$$\neg P(t) = (\beta, \alpha, \gamma) \text{ if } P(t) = (\alpha, \beta, \gamma) \text{ where } \alpha + \beta + \gamma = 1$$

in other words, the truth value of ¬P will be the false value of P and conversely, and P and ¬P, will have the same value for N.

It has been defined "$\overset{T}{\rightarrow}$" to have values 0 or 1. If we suppose that it takes intermediate values, we will generalize the temporary link notion to temporary link in a degree α. It is written:

$$\Big\|\overset{t}{-}_{\alpha}$$

and has:

$$P(t)\Big\|\overset{t}{-}_{\alpha} Q(t) \quad \text{if} \quad \Big\|\overset{t}{-}_{\alpha} P(t) \rightarrow Q(t)$$

5 Dynamic Fuzzy Rules of Inference

We will develop other ideas on dynamic fuzzy logic as the conditional dynamic fuzzy propositions, and the dynamic compositional rules of inference.

In classic propositional calculus we consider the expression "if A then B": $A \rightarrow B$ as:

$$A \rightarrow B = \neg A \vee B$$

A dynamical fuzzy conditional A (t) $\overset{T}{\rightarrow}$ B (t), with A(t) as antecedent and B(t) as consequent, refer to fuzzy sets instead of propositional variables.

Examples of this are: If slippery then hazardous in rainy time. If the highway is slippery today, then it is hazardous to drive.

In essence these propositions describe relationships among two dynamic fuzzy variables. Thus the conditional dynamic fuzzy propositions are defined as fuzzy

relationships, and not as logical connectives. For this is defined the cartesian product of two dynamical fuzzy sets as following:

Let A(t) a fuzzy subset of the universe U(t), and B(t) a fuzzy subset of another different universe of the discourse, then the cartesian product of A(t) and B(t) is defined by:

$$A(t) \overset{T}{\times} B(t) = \int_{U \times V} \psi_{A(t)}\big(U(t)\big) \overset{T}{\wedge} \psi_{B(t)} / \big(U(t), V(t)\big)$$

The meaning of a dynamical fuzzy conditional of the form A(t) then B(t), is clarified considering that it is a particular case of : if A(t) then B(t), otherwise C(t) in a time t. Where A(t), B(t), C(t), are fuzzy subsets of the universes of discourse U(t), V(t).

The consideration of $C(t) \in V(t)$ implies that in absence of an indication of the contrary, the consequent of:

$$\overset{T}{\neg} A(t) \overset{T}{\rightarrow} C(t)$$

can be any dynamical fuzzy subset of the universe of discourse.

The dynamic fuzzy inference rule will be defined in the next form: If R(t) is a dynamical fuzzy relationship of U(t) to U(t), and X(t) is a fuzzy subset of U(t), then the fuzzy subset Y(t) of U(t), that it is induced by X(t) is given by the composition:

$$Y(t) = X(t) \overset{T}{\circ} R(t)$$

in which:

$$X \overset{T}{\circ} R = \int_{U(t) \times V(t)}^{T} V_{y(t)} \left[\left(\psi(t) \overset{T}{\times} \big(U(t), V(t)\big) \overset{T}{\wedge} \psi(t) R(t) \big(U(t), V(t)\big) \right) \right] / \big(U(t), V(t)\big)$$

and $\overset{T}{\times}$ plays the paper of an unitary relationship. It can be seen as a dynamic generalized Modus Ponens.

6 Dynamic Linguistic Reasoning

We will define a dynamic linguistic variable as a variable whose values are words or sentences in a natural or artificial language whose meanings change in the time. For example AGE is a linguistic variable if its values are linguistic and not numerical, for example: young, very young, etc., and they are changing with the time. In terms more formal: A dynamic linguistic variable is characterized by (X(t), T(t)(X), U(t), G(t), M(t)) where: X(t) is the name of the variable. T(t)(X) is the set of their linguistic values. U(t) is the universe of the discourse. G(t) is the syntactic rule that generates the terms in T(X). M(t) Is the semantic rule that associates to each linguistic value x, its meaning M(X), where M is a fuzzy subset of U.

The meaning of a linguistics value x (t) is characterized by a compatibility function C(t), C(t): $\longrightarrow$ [0,1], that associates with each u(t) in U its compatibility with x(t). The function of the semantic rule is to relate the compatibility of the primary terms to a compound linguistics value that changes in the time. The connectives (and, or) will be considered as non-linear operators that modify in the time the meaning of their operands in a defined form. We will consider the linguistic values as expressions of the form: h(t) U, where h(t) is a limit on the dynamic linguistics, and U is a primary term. In union with the logical connectives, (and, or) and the dynamic linguistic limitants, the primary terms serve as the generators, as the values as a dynamic linguistic variable, constituting the sense of each value in the time, a dynamical fuzzy subset of U. For example: the truth value of the linguistic dynamical term "True" can be represented as the dynamical fuzzy subset:

$$\text{True} = \int_0^1 \psi(t)\,\text{True}\,(V(t) / V(t)$$

where the integral indicates the union of the elements: y(t) True (v(t)) / v(t) oscillating v in the interval [0,1].

If "True" is defined in that form, "not True" will be defined as:

$$\text{not True} = \int_0^1 (1 - \psi(t))\,\text{True}\,(V(t) / V(t))$$

and "Untruthful":

$$\text{Untruthful} = \int_0^1 \psi(t))\,\text{True}\,(1 - V(t) / V(t))$$

With these elements is created what we will call dynamical approximate reasoning.

" The dynamic fuzzy logic is the base of the dynamical approximate reasoning, that it is a new manner of reasoning, in which the truth values and the inference rules are fuzzy, and change over time ".

We will substitute the term "statement" or "dynamic statement" for the term "proposition". A statement has the form: O(t) is A(t) in a time t, where O(t) is the name of an object, and A(t) is the name of a possible dynamical fuzzy subset of the universe of the discourse U. If we interpret A(t) as a fuzzy predicate, then the statement: O(t) is A(t), in the time t, can be paraphrased as: O has the property A in the time t. Equivalently "O(t) "is" A(t) ", can be interpreted as an assignment equation in which a dynamical fuzzy set A(t) is assigned as a value to a dynamic linguistic variable that denotes an attribute of O(t) in the time t. This statement is associated with two dynamical fuzzy subsets:

1. The meaning of A(t), M(t)(A) that it is a fuzzy subset of U(t).
2. The truth value of " O(t) "is" A(t) ", denoted by V(A(t)), and is defined as a possible subset of a universe of truth values.

We said that *Truth* is the name of a dynamic linguistic variable in which the first term is *True*, with *Untruthful* as the negation of *True*, thus:

$$\psi(t)\,\text{Untruthful}\,(V(t)) \overset{T}{=} \psi(t)\,\text{True}\,(1 - V(t) \quad \text{with } 0 \overset{T}{\leq} V(t) \overset{T}{\leq} 1$$

To build the base of a dynamic fuzzy logic, is extended the meaning of the logic operations $\vee$, $\wedge$, to operate with linguistic dynamical values instead of numerical. Let A(t) a fuzzy subset of the universe of discourse U(t):

If V_T (A(t)) is a point in V_T = [0,1], representing the truth value of the proposition A(t), the truth value of $\neg$A(t) will be:

$$V_T\left(\neg \overset{T}{A}(t)\right) = 1 - V_T(A)$$

Let V_T(A(t)) and V_T(B (t)) the truth linguistic values of the propositions A_T(t) and B_T(t). Then we can write:

$$V_T\,(A(t)) \overset{T}{\wedge} V_T\,(B(t)) \text{ for } V_T\,(A(t) \text{ and } B(t))$$

$$V_T(A(t)) \overset{T}{\vee} V_T(B(t)) \text{ for } V_T(A \text{ or } B)$$

$$V_T(A(t)) \overset{T}{\longrightarrow} V_T\,(B(t)) \text{ for } V_T(A(t) \longrightarrow B(t))$$

$$\overset{T}{\neg} V((At)) \text{ for } V_T\left(\neg \overset{T}{A}(t)\right)$$

where $\overset{T}{\wedge}$ it is the min operator in a time t (conjunction) and $\overset{T}{\vee}$ it is the max operator in a time t (disjunction).

The truth value of A $\overset{T}{\rightarrow}$ B, depends on the form in which the connective" $\overset{T}{\rightarrow}$ "is defined for numerical truth values.

7 A Generalized Fuzzy Modus Ponens

The basic rule of inference in the traditional logic is the Modus Ponens explained here. According to this, we can infer the truth of a dynamic fuzzy proposition B(t) from the truth of another A(t):

$$\text{If } A(t) \overset{T}{\longrightarrow} B(t)$$
$$\text{and we state the truth of } A(t)$$
$$\text{Then we can state the truth of } B(t)$$

In the human reasoning the Modus Ponens is used more in an approximate form instead of the exact form. Typically we know that if A(t) is true, and that A(t)* $\longrightarrow$ B(t), where A*(t) it is in some sense an approach to A(t), you can infer that B(t) is approximately certain.

Based on this, a composicional inference rule can be defined that is only a generalization of the following thing: Suppose that one has the curve y = f(x), and x = a is given. Then one can infer y = b = f(a).

If we generalize this process considering that x is in an interval, and that f(x) it is a function valued in the interval, then to find the interval y = b that corresponds to the interval a, we must first to build a cylindrical group with base in a, and to meet its intersection I with the curve valued in the interval. Then the intersection in the axis x is projected, finding the wanted y in the interval b.

Let U and V two universes of discourse with variable bases u(t) and v(t) respectively. Let R(u(t)), R(u(t), v(t)) and R(v(t)) dynamic fuzzy restrictions in the understanding that they are relationships in U, U × V, and V. Let also A(t) and F(t) particular fuzzy subsets of U and U × V, then the composicional inference rule says that the solution of the relational assignment equations:

$$R(t)\ (U) \overset{T}{=} A(t)$$

and

$$R(t)\ (U,V) \overset{T}{=} F(t)$$

are given for:

$$R(t)\ (V) \overset{T}{=} A(t)\ o\ F(t)$$

In this sense one can infer from $R(t)\ (u) \overset{T}{=} A(t)$ and $R(t)\ (u,v) \overset{T}{=} F(t)$ following:

$$R(t)\ (v) \overset{T}{=} A(t)\ o\ F(t)$$

In this way we can see the Modus Ponens as a special case of the composicional inference rule.

In traditional logic, the material implication is defined as a logical connective for propositional variables. If A(t) and B(t) are two propositionals variables, then $A(t) \overset{T}{\rightarrow} B(t)$, at a time t. In many human discourses we have that A(t) and B(t) are dynamic fuzzy sets or dynamic fuzzy predicates, instead of propositional variables. To extend the notion of material implication to dynamic fuzzy sets, let U and V two possible universes of the discourse, and let A(t), B(t) and C(t) fuzzy subsets of U,V and V. First will be defined the meaning of the expression ***if A(t) then B(t) otherwise C(t),*** and then it will be defined ***if A(t) then B(t)*** as a particular case of the previous one. The expression ***if A(t) then B(t) otherwise C(t)***, it is a fuzzy binary dynamic relationship defined in UxV for:

$$\textit{if A(t) then B(t) otherwise C(t)} \overset{T}{=} A(t) \overset{T}{\times} B(t) + \overset{T}{\neg} A(t) \overset{T}{\times} C(t)$$

For the case *if A(t) then B(t)* it is considered that $C(t) \overset{T}{=} V$ and:

$$\textit{if A(t) then B(t)} \overset{T}{=} A(t) \overset{T}{\times} B(t) + \overset{T}{\neg} A(t) \overset{T}{\times} B(t)$$

Based on the above it can be defined the Modus Ponens in the following way:

Let $A_1(t)$, $A_2(t)$ and B(t) fuzzy subsets of U, U and U ×V respectively. If $A_1(t)$ is assigned to the restriction R(t)(u) and the relationship

$$A_2(t) \overset{T}{\rightarrow} B(t)$$

is assigned to the restriction R(t)(u(t),v(t)). Then:

$$R(t)(u(t)) \overset{T}{=} A_1(t)$$
$$R(t)(u(t),v(t)) \overset{T}{=} A_2(t) \overset{T}{\rightarrow} B(t)$$

As we see this assignment relacional equation can be solved for the restriction in v(t), obtaining:

$$R(t)\ (v(t)) \overset{T}{=} A1(t)\ o\ (A2(t) \overset{T}{\rightarrow} B(t))$$

An expression for this inference has the form:

$$A_1(t) \text{ premise}$$
$$A_2(t) \overset{T}{\rightarrow} B(t) \text{ Implication.}$$
$$A_1(t)\ o\ (A_2(t) \overset{T}{\rightarrow} B(t)) \text{ Conclusion.}$$

and it constitutes the establishment of a generalized Modus Ponens. This differs of the traditional Modus Ponens in two aspects:

1. $A_1(t)$, $A_2(t)$, and B(t), are dynamic fuzzy subsets.
2. $A_1(t)$ should not be identical with $A_2(t)$.

8 Dynamic Approximate Reasoning

With these developments of the fuzzy logic is defined the dynamic approximate reasoning, as a reasoning form that implies the following fundamental elements:

1. The employment of linguistic truth values depending on time to express the degree of fuzzy asseverations and their combinations.
2. The representation of fuzzy asseverations as dynamic relational equations.
3. The solution of assignment relational dynamic equations, for the fuzzy restrictions on specific variables depending on time.
4. An approach to the solution of assignment relational dynamic equations for dynamic linguistic values.

The employment of the dynamic fuzzy logic for the handling of dynamic uncertain concepts, allows us to consider certain elements that can not be operated with effectiveness or correctly with conventional techniques. The most important points in these elements are:

The fuzziness of the antecedents and consequent in rules of the form:

1. if x(t) is A, then y(t) is B
2. if x(t) is A, then y is B, with certainty factor $\overset{T}{=} a$

Where the antecedent "x(t) is A" and the consequent "x(t) is B" are dynamic fuzzy propositions, and **a** is a numeric value of the certainty factor, for example:

$$\text{If x(t) is small, then y(t) is long, with C.F.} \overset{T}{=} 0.8$$

in the which the antecedent "x(t) is small" and the consequent "y(t) is long" are dynamic fuzzy propositions, since the denotación of the predicates, small and long, are dynamic fuzzy subsets.

3. The presence of dynamic fuzzy cuantificators in the antecedent and/or consequent of a rule, such as: several, few, many more, usually, etc. Like an illustration let us consider a dynamic disposition as the following one: (disposition is a proposition that implies dynamic fuzzy cuantificators)

"The students in primary are children" that can be interpreted as "Many students in primary are children". This in turn can be expresed like a conditional proposition: "If x(t) is a primary student, then it is possible that x(t) is young", in which "is possible" has the same denotation as a dynamic fuzzy subset of the unitary interval that the cuantificator "many".

References

Pérez-Silva J L, A. Miranda, A. Garces "A Short Term Memory Phenomenon in a Single Neuron" Advances in Artificial Intelligence and Engineering Cybernetics., Editor George E. Lasker. Published by International Institute for Advanced studies in Systems Research and Cybernetics. Vol lX Pag. 6-10. 1999.

Pérez-Silva J.L., Lara-Rosano F., Herrera A., Bañuelos M., Quintana S., Castillo J., Martínez W., Padrón A., Garces A., Prieto R.. "Dynamic Approximate Reasoning". Advances in Artificial Intelligence and Engineering Cybernetics. Editor George E. Lasker. Published by International Institute for Advanced studies in Systems Research and Cybernetics. (Accepted).

Pérez-Silva J L, Lara-Rosano F, A. Mirada, A. Herrera, S. Quintana, M. Bañuelos, W. Martínez, J. Castillo, A. Padrón, A. Garcés. "Dinamic Fuzzy Sets". Advances in Artificial Intelligence and Engineering Cybernetics. Editor George E. Lasker. Published by International Institute for Advanced studies in Systems Research and Cybernetics. Vo lII, Pag. 39-43 . 1998.

Generation of a Personal Qualitative Assessment Instrument Using a Fuzzy Expert System

A.M. Martínez-Enríquez and O.R. Sereno-Peñaloza

Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN
ammartin@mail.cinvestav.mx

Abstract. We present an expert system that generates a personal assessment instrument to evaluate the attitude or tendency of a person who wants to realize a particular activity. For example, suppose a student wants to carry out a postgraduate course. In order to evaluate the possibility of the student's success, it seems desirable to know (among others aspects) the priority the student gives the course with respect to any other activity.
Our expert system generates a personal test to evaluate three classes of diagnostic areas:
- antagonistic behavior (such as autonomy and dependence);
- predilection toward different interests (e.g. personal realization in preference to professional development);
- equilibrium between demand (e.g. by a student from the institution) and commitment to realize an activity. We formalize the criteria and strategies used by evaluating each class of diagnostic areas. Fuzzy logic is used to select the questions that compose the test, and the way in which the system produces a personal assessment instrument. Our expert system has been validated to evaluate the attitude of candidates who desire to pursue a postgraduate program.

1 Introduction

The construction of instruments of assessment has been a matter of interest for much time, some examples [Aiken 79] are: In 2,200 BC Mandarins set up a civil service testing program in China; in 1200 AC, the first formal oral examinations in law were held in the University of Bologna. At the end of the last century, the aim of the philosopher was to establish a relationship between the mind and the world, as well as the way in which the physical intensity was related with the psychological aspect. For instance, they tried to evaluate the highly skilled, as budgie–feet measures the intensity of the light, or like sound by decibel. Hence, one of the first instrument of assessment named „distinguish perceptible differences" [Thorndike 14], makes people listen two sounds, which are slight different, in order to differentiate their intensity; the grade of confusion is a function of the number of mistakes. This is highly psychological [Robyn 75]. By contrast, Thurstone claimed that this measure could be uniquely determined by a probability distribution function. Thus, Thurstone

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 671—686, 2000.

[Thorndike 41] constructed an instrument of assessment for measuring attitudes, named the Law of Comparative Judgment, by which several judges are asked to sort into 11 categories a large number of sentences, expressing positive or negative attitudes toward groups, institutions, or concepts. The ranging is from least favorable (category 1) to the most favorable (category 11).

By the 1960s the use of computers was increased in designing, administering, scoring, analyzing and interpreting tests, but this usage was normally directed to simplify the task made by the administrator. The instrument's orientation had not been modified. Those computer systems are static, i.e. if a new model of instrument rises or its approach changes, it will be necessary to construct a new computer program.

Our instrument of assessment is constructed as a system based upon rules, written in FCLIPS (Fuzzy C Language Integrated Production System) [FCL, STB 97]; the knowledge base is an inquiry, each question is represented by rules. The plan and strategies followed by the system are according to each class of diagnostic area. In this way, the instrument of evaluation uses fuzzy logic [Zadeh 65, Zadeh 83], which is applied successfully on imprecise concepts, like qualitative abilities and attitudes. We also introduce other new techniques for modeling inquiries.

In the majority of the inquiries, the following particularities are observed: -The use of multiple choice questions, like in Osiris Trivia Quiz [Osiris], Micromedium [MPC], retina quiz [Yichie]. -Each question is independent of any other included in the test. -Only the number of right answers is obtained as a result of the evaluation. However, the correctness of qualitative abilities and attitudes, and the meaning of „right" are difficult to define. Thus, we propose a method to automatically generate tests for qualitative attitudes and strategies to evaluate different classes of them by means of an expert system.

The main problem in evaluating the qualitative concepts is their imprecision. Yet fuzzy theory [Zadeh 65, Zadeh 83] has been successfully applied on imprecision using linguistic variables. Similarly, our system attaches a linguistic label to each alternative response of a question, in order to validate the answer and to personalize the instrument of assessment.

The rest of the paper is organized as follows: In section 2, we define the general problem, presenting the characteristics of the diagnostic areas, and classifying them. Each class is formalized, as well as, it is described its strategies and its algorithms to asses each class. In section 3, we analyze the results of some studied cases in order to validate our proposal. In section 4, we describe the knowledge base of our expert system. In section 5, we conclude and discuss further research.

2 The Problem Definition

The aim of our instrument of assessment is to serve practical ends, such as knowing the profile of the examinee, his beliefs and mental impression about the institution, where he desires to be enrolled, and which aspirants might achieve their ambitions with success. In addition, a well instrument of assessment must be as personal as individuals exist, thus the inquiry must be constructed out of inconsistency, whose

strategies of solution will be ad hoc and can be performed automatically. Hence, the general problem in our system is defined as follows.

Given:

A set of diagnostic areas $\Omega=\{E_1, E_2, ..., E_s\}$, and $\Delta_i \subset \Omega$, $1 \leq i \leq n$, where Δ_i is a specific class of diagnostic area, such that class(Δ_i) $\in \{\Delta_1, \Delta_2, \Delta_3\}$.

Where:

Δ_1: subset of diagnostic areas, representing antagonistic behavior. For instance, people are either autonomous or dependent, but in any case both qualities are presented at the same time. This class is named **Differential Semantic**.

Δ_2: Subset of diagnostic areas, which are susceptible to be related, and to establish a priority among them. For instance, people realize an activity, because either their professional experience is improved, or their income is increased. This class is named **Relationship of Precedence**.

Δ_3: Subset of diagnostic areas that finds out the exigency and the commitment within different situations. This class is named **Equilibrium**.

Goal: - To generate a personal inquiry out of inconsistency, whose strategies of solution for each class will be ad hoc.
- To define the profile of the examinee.

In order to reach the goal, our system carries out a plan of strategies trying systematically each class in a homogeneous framework. This means that an inquiry includes questions with multiple choice as a common technique, since it is easy to evaluate them. However, each alternative answer is associated to a linguistic term, which is a function of the class, as explained in sections 2.1, 2.2, and 2.3.

In order to define the profile of the examinee, our system uses a set of linguistic terms to diagnose the aptitude and disposition of a candidate. The set is:

$$\boldsymbol{LT} = \{\text{Null, Slight, Moderate, Almost, Complete}\}.$$

The result of a diagnosis represents the awareness (resp. ignorance) about the degree in which a person is placed in a specific diagnostic area (e.g. security-insecurity):

$$\textbf{Awareness of } E_i \rightarrow \{\text{Null, Slight, Moderate, Almost, Complete}\}$$

Respectively, linguistic term $\boldsymbol{ut} \in \boldsymbol{LT}$ concerning the **unawareness** of E_i is calculated by:

$$\boldsymbol{ut} = r - at + 1,$$

Where r is the number of linguistic terms of $\boldsymbol{LT}$ (five for this case),
at is the position of the linguistic term into $\boldsymbol{LT}$, concerning awareness of E_i.

2.1 Differential Semantic Area

The purpose of the questions that are related to the differential semantic is to determine the tendency of the examinee in one of the antagonistic behaviors. For example, consider the following question:

Select two statements from the following declarations, which you agree with the most:

- Some people:

(a) Express their opinion at last.

(b) Argue convincingly, when they discuss.

(c) Speak in public, because they consider that as positive action.

(d) Think about possible critics.

(e) Do not have problems to have a conversation with anyone.

The above question tries to evaluate security (S+) or insecurity (S-) tendency. Thus the options (b) and (e) are related to security (b, e $\in$ S+); (a) and (d) show insecurity (a, d $\in$S-), and (c) is a neutral alternative.

Inquiring people are usually perspicacious. Hence it is difficult to be sure about the correctness and the validity of the inquiry. In order to make a good diagnosis within differential semantic class, we formulate the following strategy:

Let $\Delta_i=\{E_1, \ldots, E_i, \ldots, E_r\}$ be a set of areas, where E_i represents antagonistic behavior representing by two polarities: {E-, E+}, e.g. insecurity (-) and security (+). Each E_i is composed by a set of questions $\{q_1, \ldots, q_k, \ldots, q_s\}$, where $\mathbf{q_k}$ is modeled by a set of alternative solutions, each one related with a linguistic term, $\boldsymbol{t}\in$(A++,A+,$A_{neutral}$,A-, A--), such that (A++, A+) $\in E_i^+$ y (A-, A--) $\in E_i^-$, and $A_{neutral}$ is a neutral alternative.

In order to determine the ***tendency*** of each antagonistic behavior of examinee, each question **qk** is answered by selecting two alternatives r=(r1, r2). Such that, the first selection has priority over the second one: r1>r2. By this selection, it is possible to validate the answer, as we see in the following cases:

a) If r1 and r2 $\in$(E+) (resp. E-),
then confirm **T**endency of one polarity: Diagnosing $(q_k)\leftarrow$**T+** (resp. **T-**)

b) If r1 and r2 $\in$ (E+ $\cup$ $A_{neutral}$), (resp. (E- $\cup$ $A_{neutral}$)
then confirm **P**artial **T**endency: Diagnosing $(\mathbf{q_k})\leftarrow$**TP+** (resp. **TP-**)

c) Otherwise Diagnosing $(q_k)\leftarrow$ **Inv**alid

Traditional inquiries evaluate this kind of question, demanding to examinee to order the alternatives of the question, starting with the most preferred. The inquirer supposes that options of the same polarity will be placed in the extreme of the ordered set of solutions. But in reality, this is the ideal case and usually occurs a few times. In general, the inquirer must analyze all different answers given by the examinee. This task of analysis is easier than considering all possible solutions, since, as it is well known, the combination of **n** options is the factorial of **n** (120 for the present case). On the other hand, if two choices from **n** alternatives exist, the number of possible combinations is n! /((n-k)! * k!); i.e. 10 for the present case. Hence, when the number of questions is relatively large, the diagnosis can be realized in a subjective way.

In contrast, the proposed methodology considers only five possibilities for the three above situations (a, b, c). Consequently, the task of analysis is eliminated and it is possible to individualize the inquiry, inasmuch as it is not necessary to formulate all the questions, as we discuss below.

2.1.1 Plan of the Strategy

Usually, an inquiry is constructed to be completely answered. Nevertheless people with a well-defined character reveal their personality in the beginning of the answers, which must be consistent. Thus, our heuristic considers that when the early two questions (q_1,$q_2 \in E_i$) confirm the same case (a or b or c), and the same polarity (E+ or E-), this diagnostic area E_i should not be investigated any further; otherwise the system analyzes all the questions included in E_i. The score of answers is reported as the awareness of E_i by a linguistic term $\boldsymbol{t \in LT}$ =(Null, Slight, Moderate, Almost, Complete). Hence, the suggested plan of the strategy is as follows:

If $\forall q \in E_i$, {q1,..., qk}, k ≥2,
and Diagnosing (q) ∈{T+, TP+} (resp. {T-, TP-})
then Begin
Conclude Awareness of (E_i) ← E+ ***Complete*** (resp. E_i-)
Eliminate E_i from Δ_i: $\Delta_i \leftarrow \Delta_i \setminus E_i$ („\" difference within sets)

End Else

If $\forall q \in E_i$, {q1,..., qk}, k ≥ 2,
and Diagnosing (q) = **Inv**alid
then Begin
Conclude Awareness of (E_i) ← **Null**
Eliminate Ei from Δ_i: $\Delta_i \leftarrow \Delta_i \setminus E_i$
End
Otherwise Compute Score of Tendency (Ei) → {T+,TP+, **Inv**alid, TP- T-}

2.1.2 Algorithm of Differential Semantic

The principle of the algorithm is simple. All diagnostic areas $E \in \Delta_i$ are tested, when a certain number of questions pertained to E have the same tendency (E+ or E-), the system excludes the rest of the questions pertained to this area E, since it considers that the personality of examinee is defined. Thus, an exhaustive inquiry is not necessary. The remaining diagnostic areas ($\Delta_i \leftarrow \Delta_i \setminus E$) are submitted to the same process.

Differential_Semantic (Δ)
Begin
i←0; //diagnostic area in treatment
Status (inquiry)←ϕ; //Score of each question
Tendency (Δ)←ϕ; //Tendency of Ei={**Inv**, T+, TP+, T-, TP-}
∀Ei∈ Δ Do begin
i ← i+1; // related to Ei
Tps←0; Pps ← 0; Tng ← 0; Png ← 0; Invalid ← 0;
Tendency (i) ← ϕ; Status (i) ← ϕ;
k ←0; // related to qk
m ← |Ei | // control Ei
While Ei ≠ ϕ and k ≤ m

```
        Do Begin                                    //select q_k ∈ Ei
                k ← k+1; q ← qk
                Ei ← Ei \ qk                        // update Ei
                inquire (q);
                if  k ≥2 and Invalid ≥k
                then Begin                          // Ei is invalid
                     Tendency (i)← Tendency (i) ∪ (Inv, Invalid/k);
                     k←m +1                         //stop Ei
                 end Else
                if  k ≥ 2 and ( Tps +  Pps ) ≥ k
                then Begin                          // Ei+ is confirmed
                      Tendency(i)←Tendency (i)∪ ((T+, Tps/k)(TP+, Pps/k));
                      k←m+1;
                end Else
                if  k ≥ 2 and  ( Tng +  Png ) ≥ k
                then Begin                           // Ei- is confirmed
                     Tendency(i)←Tendency (i)∪((T-,Tng/k)(TP-, Png/k) );
                      k← m+1;
                end;
        end;                                        //end of treatment of Ei
        if Tendency (i)=ϕ
        then Begin
                Tendency (i)←(Inv, Invalid/m)∪(T+,Tps/m)∪(TP+,Pps/m)∪
                                        (T-, Tng/m) ∪ (TP-, Png/m)
         end;
        Tendency (Δ)← Tendency (Δ)∪ Tendency (i);
    end∀
    Return (Tendency (Δ), Status (inquiry))
End
inquire (q)
Begin
    if  (r1 and r2) ∈(A+, A++)                                      //T+
    then        Tps ← 1+ Tps;
    Else  if (r1 and r2) ∈ (A-, A--)                                //T-
          then      Tng ← Tng + 1
    Else  if  (r1 and r2 ) ⊂ (A+ ∪ A++ ∪ Aneutral)                  //TP+
          then      Pps ← 1+ Pps
    Else  If (r1 and r2) ⊂ (A- ∪ A-- ∪ Aneutral)          //TP-
          then      Png ← 1+ Png
    Else  Invalid ← Invalid +1;                                     //Inv
          status (inquiry) ← status (inquiry) ∪ (q, r1,r2)
End
```

The score of the tendency of E is treated as follows:

Let Tendency (E)={(Inv, I)(T+, Tps)(TP+, pps) (T-, Tng)(TP-, png),

Where:
I = number of invalidated questions divided by the number of proposed questions of E.
Tps= number of confirmed E+ divided by the number of proposed questions of E.

pps= number of partially confirmed E+ divided by the number of proposed questions E

Tng = number of confirmed E- divided by the number of proposed questions of E.

png = number of partially confirmed E- divided by the number of proposed questions E

In order to simplify the diagnosis, the tendency T+ is added with the partial tendency TP+ (resp. T- with TP-), thus Tendency (E) is transformed as:

$$\text{Tendency (E)} = \{(\text{Inv, I})\ (\text{E+, P})\ (\text{E-, N})\}$$

Where:

$$P = Tps + Pps; \qquad N = Tng + Png$$

As antagonistic situations cannot be present in the same situation, each confirmed answer E+ invalidates other one confirming E-, and those ones are added to the invalidated answers:

If $N \geq P$

Then begin $N \leftarrow N\text{-}P;\ I \leftarrow I + 2*P;\ P \leftarrow 0;\ \text{Score} \leftarrow \{I, N\}$ end

Else begin $P \leftarrow P\text{-}N;\ I \leftarrow I + 2*N;\ N \leftarrow 0;\ \text{Score} \leftarrow \{I, P\}$ end

Each element $\boldsymbol{e} \in \text{Score} \rightarrow [0,1]$, and $\sum \boldsymbol{e} = 1$. In addition, the number of questions for each area is odd and greater or equal to the number of possible solutions (five for the present case). As a result, a maximum $\boldsymbol{e}$, $\boldsymbol{e} \in$ Score exists, as it is deduced by the above conditions. The score only considers two possible cases to specify the position of the examinee within E: Score ={I, (P or N)}. Where **I** is the number of invalidated questions, representing the grade of ***unawareness***, while **P** or **N** is the number of questions confirming one polarity, representing the grade of ***knowledge*** about E+ or E- respectively.

$$\text{Awareness (E)} \rightarrow \{\text{Null, Slight, Moderate, Almost, Complete}\}$$

Respectively, the position $\boldsymbol{ut}$ for unawareness (E) is $\boldsymbol{ut} = r - \boldsymbol{at} + 1$, where r is the number of linguistic terms (five in this case); ***at*** is the position of the linguistic term of Awareness (E).

The awareness of a diagnostic area is the function of the score obtained by the system, thus the membership of this score to the linguistic term is defined by a fuzzy interval. For instance, for the case of 5 questions the membership function is as follows:

$\text{Score} < 0.2 \rightarrow \text{Null},$

$0.2 \leq \text{score} < 0.4 \rightarrow \text{Slight},$

$0.4 \leq \text{score} < 0.7 \rightarrow \text{Moderate},$

$0.7 \leq \text{score} < 0.9 \rightarrow \text{Almost},$

$0.9 \leq \text{score} < 1 \rightarrow \text{Complete}$

2.2 Relation of Precedence

A challenge can be carried out by different reasons. For instance, some people choice to realize an activity because his professional development (e.g prestige) is more

important than his personal realization (e.g. income) or sometimes because they want to please relatives or friends (e.g. dependence). Hence, from this point of view, a question can be perceived as a relationship. Thus each options of a question is associated to a particular area: $(A_i \in E_i)$,..., $(A_j \in E_j)$, j>2, where $E_i,...,E_j \in \Delta_2$, and A∈Neutral. For example:

Express your opinion about the following statement by selecting two alternatives, where the first one has priority to the second:

I think the principal reasons to pursue a postgraduate degree are:

(a) To know best my profession
(b) To increase my income
(c) To have the possibility to use the Internet
(d) To update my knowledge
(e) To have the same status as my colleagues
(f) To follow something different
(g) To claim a good cultural level
(h) To please my family

Where:

(a, d)∈ ***professional*** development, (b, g)∈ ***personal*** realization,
(e, h)∈ ***dependence***, and (f, c)∈ ***neutral***.

For this class of diagnostic areas, the problem is defined as follows:

Given: $\Delta_2 = (\cup_{i=1,\, s} E_i, >)$, s>2 a set of areas, such that E_i has precedence > over E_j, E_k. This precedence will be manifested by selecting two options with priority r=(r1, r2), r1> r2 for answering each question qk, qk∈ Δ_2. Each alternative is associated to a linguistic term: {$((A_i++, A_i+) \in E_i)$, $((A_j++, A_j+) \in E_j)$, $((A_k++, A_k+) \in E_k)$, and (A++, A+)∈ Neutral}.

Goal: to establish the precedence among different areas.

The strategy of solution considers three possible situations:

a) If $(r_1, r_2) = (A_i++, A_i+) \in E_i$ (resp. $(A+, A++) \in E_j$ or $(A+, A++) \in E_k$),
then $\mathbf{E_i}$ has **priority** over E_j and E_k
if the early two questions confirm the same priority (e.g. $E_i > E_j, E_k$),
then - to eliminate all alternatives pertained to E_i: $(A_i+, A_i++) \in Ei$ from the rest of the questions concerning Δ_2;
- to follow the inquiry relating only $(E_j, E_k,...) \in \Delta_2$, in the same way

b) If $r_1 \in (A_i+, A_i++) \in E_i$ and $r_2 \in (A_j+, A_j++) \in E_j$, (resp. (E_i, E_k) or (E_j, E_k))
then $\mathbf{E_i}$ has **priority** over E_j, and $E_i, E_j > E_k$ (resp. $E_j > E_i$ or $E_k > E_i$ or $E_k > E_j$,...)
if two consecutive questions confirm the same priority (e.g. $E_i > E_j$)
then - to eliminate all alternatives pertained to E_i: $(A_i+, A_i++) \in E_i$, from the rest of the questions qk∈ Δ_2 ($\Delta_2 \leftarrow \Delta_2 \setminus E_i$)
- to follow the same strategy, only with questions relating the other areas, i.e. $(E_j, E_k,...) \in \Delta_2$

c) If $r \subset A_{neutral} \cup E_i$, (resp. $A_{neutral} \cup E_j$ or $A_{neutral} \cup E_k$),
then $\mathbf{E_i}$ **has** priority **P**artially confirmed over E_j and E_k
if three consecutive times E_i presents the same priority,

then - to eliminate all alternative $(A_i+, A_i++) \in E_i$;
- to follow the same strategy with questions including only alternatives pertaining to $(E_j,E_k,...) \in \Delta_2$

d) Otherwise qk is invalidated (i.e. $r \in A_{neutral}$)
if three consecutive times the qks are invalidated,
then Awareness (Δ_2) ← **Null** (resp. complete unawareness)

Traditional inquiries try to evaluate the precedence among diagnostic areas by asking inquiring people to sort in decreasing order the alternatives of the question. Suppose that we have three areas to be related. In such a case, a question is modeled by two alternatives for each area Ei, and two for Neutral. Hence, in the worst-case, the number of possible solutions that the inquirer could analyze is 8! =40320. By contrast, our strategy of solution has 13 possible responses: - three different cases for (a): $\{(E_i>E_j,E_k), (E_j>E_k,E_i), (E_k>E_j,E_i)\}$, - three similar situations for (c), - six for (b):$\{((E_i>E_j$, and $E_i,E_j>E_k)$, $(E_j>E_i$, and $Ei,Ej>E_k))$, (resp. $(E_i>E_k$, and $Ei,E_k>E_j)$, and $(E_j>E_k$, and $E_j,E_k>E_i)\}$, as well as, -one for (d). In order to be coherent for this strategy, the system must include 13 questions as a minimum.

Normally, the same case (a or b or c or d) does not happen consecutively, hence the score of different cases is evaluated as follows:

a) If $E_i>E_j,E_k$, then priority (E_i)← priority (E_i) + 3
b) If $(E_i>E_j$, and $E_i,E_j>E_k)$,
Then begin priority (E_i)← priority (E_i) + 2; priority (E_j)←priority (E_j) +1 end
c) If $(E_i>\boldsymbol{p}\ E_j,E_k)$, (>**p**: partially priority)
Then begin priority (E_i)← priority (E_i) + 1;
priority $(A_{neutral})$←priority $(A_{neutral})$ +1 end
d) If $r \notin (E_i,E_j,E_k)$ then priority$(A_{neutral})$←priority$(A_{neutral})$ +6

In this way, if the four above cases occur with all their possible alternatives, each one will have the same value of priority; this situation represents the uncertainty of knowledge about preference of the examinee with respect to this class of diagnostic areas (Δ_2). In addition, as we see above, the strategy considers that if three consecutive questions are invalidated (case d), the ***Awareness*** about preference of areas from the examinee is Null (resp. complete unawareness).

As the result of diagnosing, the system presents an ordered set of areas, e.g. $(E_i>E_j>E_k)$, taking account of the value of priority obtained, or a complete unawareness in the case of invalidated questions (priority$(A_{neutral})$ >> $\forall E_i, E_i \in \Delta_2$.

2.3 Equilibrium between Exigency and Commitment within Different Situations

Commonly, people demand from others when they don't have a real sense of commitment towards others. For instance, students demand prerogatives from an educational institution, but not all of them want to be engaged with the institution. An example of a question that tries to evaluate demand and commitment is:

Choose two options, where the first one has preference over the second:

If you were accepted into this postgraduate program, what would you wish that the Institution had?

(a) A well-equipped and stimulating environment, in order to apply our knowledge and improve our experience.
(b) A good sportive installation, and green spaces.
(c) Excellent staff of teachers, who make us participate in innovative research projects having, in this way, a compromise to make good teamwork.
(d) A highly motivated staff of researches, who gives us their background knowledge, as well as experiences.
(e) A group for sharing knowledge and experience, in order to improve our knowledge, and to have little by little more responsibility in our professional formation.

Where: (a, d) ∈ Exigency (-), (c, e) ∈ Commitment (+) y (b) ∈ Neutral.

The definition of the problem is as follows:

Given $\Delta_3=\{q_1,..., q_k,..., q_n\}$a set of questions, such that q_k is modeled by equal number of positive (commitment) and negative (exigency) alternatives, and a neutral alternative. In order to be coherent with the above cases, a question has five options (Ac++, Ac+, $A_{neutral}$, Ae-, Ae--,), and is answered by selecting two alternatives. The goal is determine, within different circumstances, the relationship between commitment and exigency.

As a result of the diagnosis three cases are considered: -equilibrium, when the examinee chooses equal number of alternatives representing commitment and exigency; -engagement (resp. exigency), when the number of alternatives of commitment is greater than demands.

a) If | response (q) | ∈ (A+, A++) = | response (q)| ∈ (A-, A--),
Then Diagnosing (Δ_3) ← **Equilibrium**

b) If | response (q) | ∈ (A+, A++) > | response (q) | ∈(A-, A--),
Then Diagnosing (Δ_3) ← **Engagement** (i.e. Commitment > Exigency)

c) If | response (q) | ∈ (A+, A++) < | response (q) |∈(A-, A--),
Then Diagnosing (Δ_3) ← **Exigency** (i.e. Demands > Commitment)

Traditional inquiries diagnose this attitude asking people to order the options of the question according with their preference. In the ideal case, commitment will be opposed to exigency, but in reality both qualities are combined. The number of possible solutions is 120. By contrast our proposal only has 3 possible solutions.

3 Validation of the System

In two consecutive scholarship periods, the inquiry has been applied to different groups of aspirants, who want to follow a post-graduated program (computer science, communication, biolectronic, automatic control, etc.), see Table No.1. In the second one, some of those groups were students that had been inquired in the first period, then the same inquiry was applied to them, in order to observe the transformation of personality or tendency toward the institution and the development of their aims. The examinees were around 25 years old, and 40 % of them had worked in their professional domain. The spent time for answering the inquiry was between 6 to 35

minutes. The less time spent, the people's aims were more defined. In the first period of application, the results obtained in those different groups are more or less similar:

- 38 % presents ***almost complete*** **autonomy** to realize any activity, to be autodidactic, and to formulate any positive proposal,
- 38 % is ***slightly*** confident of one's own abilities (security),
- 30 % has ***complete*** **positive expectation** towards the institution in order to reach their postgraduate studies,
- 61.5 % claims to work together with the institution in their professional formation (***moderate*** **commitment**),
- 69.2% shows more preference for their personal realization than for their professional development.

In the second period of application, the results are something different, especially with students who are officially registered into a postgraduate program, as we observe in the table 1. Aspirants show commitment; by contrast enrolled students denote exigency. Aspirants have less expectation toward the institution to reach studies than enrolled students do.

Table 1 Results obtained by the expert system for different groups of postgraduate programs

Postgraduate **program**	**Autonomy**	**Security**	**Positive expectation**	**Exigency vs. Commitment**	**Preferences Personal** realization **vs. Professional** development
Mathematics	44 % Slight	55% Moderate	55 % almost Complete	77 % complete Exigency	88 % almost complete Personal >Professional
Chemistry	44.4 % Null (unawareness)	55 % Slight	55 % Complete	55 % moderate Exigency	55 % slight Personal = Professional
Automatic Control	57 % Moderate	85 % Slight	71.4 % Slight	71.4 % slight Compromise	60 % moderate Personal >Professional

In order to validate the inquiry generated by our system, the examinees were inquired two times: first by 3 psychologists, who are interviewing each examinee, and latter using our test. The first inquiry may have been more complete and exhaustive. However the results obtained are more or less similar, as it can be seen in table No. 2. The scale used by the group of psychologist has the following equivalence with that used by our system:

$$(0,1, 2, 3, 4) \rightarrow (\text{Null, Slight, Moderate, Almost, Complete})$$

The table No. 2 shows very similar results obtained by the specialist and those obtained by our system. The results in bold print point out the differences between them. Those differences are always one position into the ordered set of the diagnosing scale.

Table 2. Comparison between diagnosing of the psychologist and result obtained by the expert system

Examinee	Diagnosing of psychologist				Diagnosing of the generated inquiry by ES			
	Security	Autonomy	Positive Expectation	Personal Realization	Security	Autonomy	Positive Expectation	Personal Realization
1	2	2	3	3	Moderate	Moderate	Almost	Almost
2	**2**	3	**3**	2	**Almos**t	Almost	**Complete**	Moderate
3	2	3	3	**2**	Moderate	Almost	Almost	**Almost**
4	2	**3**	3	3	Moderate	**Moderate**	Almost	Almost
5	2	2	**3**	3	Moderate	Moderate	**Complete**	Almost
6	**2**	3	**3**	2	**Almost**	Almost	**Complete**	Moderate
7	**2**	**3**	3	2	**Almost**	**Complete**	Almost	Moderate
8	**2**	4	3	2	**Almost**	Complete	Almost	Moderate

Concerning the validity of each question, the system reports those that have high frequency for neutral alternatives. For instance:

The principal expectations for people who carry out a postgraduate study are: (choose two options)

(a) **To have a hobby** (Neutral 30 % selected)
(b) **To have a diploma to increase curriculum vitae** (Dependence 30% selected)
(c) To meet a lot of interesting people (Neutral 5% selected)
(d) To have more opportunities (Personal Realization: 15 %)
(e) To be in touch with consummate professionals (Personal Realization: 10 %)
(f) To be an engaged and excellent researcher (Professional development: 5%)
(g) To feel secure about one's future (Dependence: 5 %)
(h) **To be able to take more responsibilities** **(Professional** development **0 %)**

The above question tries to evaluate preferences or aims by which the examinee decides to pursue a postgraduate degree. As we see the neutral alternative (a) is greatly selected when there are more plausible options. Thus the inquirer analyses this kind of question and he has the possibility to modify them. On the other hand, those results give more information about the group's behavior (e.g. people don't like to take responsibilities, alternative h is 0% selected).

4 Expert System as Assessment of Qualitative Attitude

The personal assessment instrument has been constructed to be:

- **Extensive,** A new class of diagnostic area may be added at any time.
- **Complete**. It is possible to add more areas into each class, as well as, to modify the questions included.
- **Reliable** with respect to the diagnosis furnished, following the same judgment in any case.

To reach those qualities our system has been constructed as a Knowledge Base System (KBS), whose base of rules (RB) constitutes the inquiry. Each class of diagnostic area is explored by a set of questions written by rules. For instance, some rules written in FCLIPS for the question trying to evaluate preference are:

```
<Syntax>
      (defrule <name of the rule> [<comments>]
            [<declaration>]                          ; properties of the rule
            <condition> *                            ;antecedent
            =>                                       ; implication
            <actions> )                              ; consequent

R1:                                                  ; Dependence
(defrule diag_dep_comb_p1 "D--, D-"
    (preg_comb_1 ?resp& h  e ) ; (h, e) = (D++, D+)
 =>
  (assert (aspirant dependent-p1)) CF 1))    ;; Assert Dependence, certainty CF=1

R2 :                                                 ; assert dependence with certainty CF=0.9
        (defrule diag_dep_comb_p1 "D--, D-"
            (preg_comb_1 ?resp& e h )                ; (e, h) = (D+, D++)
 =>
            (assert (aspirant dependent-p1)) CF 0.9) )

R3:
(defrule determina-DEP_REA-PER_ DE-PRO_P1 "precedence among Ei, Ej, Ek "
     (initial-fact)                                  ; initialization function
=>
     (display I think the principal reasons to pursue a postgraduate degree are: (choose two options)
            (a) To know best my profession
            (b) To increase my income
            (c) To have the possibility to use the Internet
            (d) To update my knowledge
            (e) To have the same status as my colleagues
            (f) To follow something different
            (g) To claim a good cultural level
            (h) To please to my family
"
            )                    ; end of the rule
                  ; In order to assure that both responses are different, the following functions are added
            (assert (preg_comb_1 (Make-question1-2
```

" What is your first selection? "
" What is your second selection? " a b c d e f g h)))) ; *end of rule*

By the above rules, it is possible to have different degrees of certainty (CF) as the user of the application wants.

In case of equilibrium between exigency and commitment, the following rules express a question and how to assure that the examinee chooses two options in response to a question.

R4:

(defrule diag_OFER_DEM_p1 "Commitment +“ ; *Equilibrium*
(preg_OF_DE_1 ?resp& **c e | e c**) ; *Responses relating to the Commitment*
=>
(assert (aspirante oferta-p1)) CF 1);

R5:

(defrule determina-OF_DE_P1 „Commitment vs. Exigency"
(initial-fact) ;*step of initialization*
=>
(display " Choose two options, where the first one has preference over the second:

If you were accepted into this postgraduate program, what would you wish that the Institution had?

(a) A well equipped and stimulating environment in order to apply our knowledge and improve our experience.
(b) A good sportive installation, and green spaces.
(c) Excellent staff of teachers who make us participate in innovative research projects having, in this way, a compromise to make good teamwork.
(d) A highly motivated staff of researches, who gives us their background knowledge, as well as experiences.
(e) A group for sharing knowledge and experience, in order to improve our knowledge, and to have, little by little more responsibility in our professional formation.

")
(assert (preg_OF_DE_1 (Do-question1-2 ***"¿Choose your first selection? "***
"¿What is the second selection? " **a b c d e**))))
; *end of rule*

5 Conclusions

This paper has presented a qualitative assessment instrument using an expert system. This instrument of assessment is a Knowledge Base System (KBS), whose base of rules constitutes the inquiry. The whole inquiry has the same framework: questions with multiple options, from which two must be selected, in order to validate them. Thus, the examinee cannot distinguish any difference.

The studied attitudes have been organized and formalized into 3 classes, susceptible to be extended, i.e. a new class can be added at any time; in such cases the system only joins new strategies written by rules. In this way, it is also possible to increase each class, as well as to modify the inquiry.

The principal aims of the inquiry are to determine:

a) The tendency under antagonistic situations, e.g. autonomy-dependence;
b) The predilection toward different interests when a specific activity is realized, e.g. professional development in preference to personal realization;
c) The current trends toward commitment or exigency when a challenge is taken.

Respecting the diagnosis furnished by our system, it is similar to those furnished by other systems. However, our approach allows a personal inquiry, because the number of proposed questions is function of the sequence of given responses. We consider that well-defined people do not need to answer a long inquiry. Moreover, when an examinee doesn't want to be inquired, the system infers this fact, and the inquiry is finished reporting **unawareness**.

Another interesting point in our proposal is that algorithms and heuristics are very simple, and they can be written either by rules or by using a specific computer language.

In order to validate our system, we selected groups of aspirants of different postgraduate studies within our institution, as a way to explore the causes of desertion. We consider that the results are very similar to those obtained by specialists in the domain, although our system can be seen as an aid-tool working always within the same judgment.

Future work will increase the classes, attitudes to be tested, as well as the availability of other heuristics in order to validate each diagnosis given by the system. In addition, some applications can be developed. For example: - A professional orientation guide, in which case, the inquiry is addressed to know the abilities and aptitudes of the examinee, in order to choose a profession. - A guide for tourists could help to propose interesting places, activities, etc. -In telephonic services to know the profile of a possible subscriber, etc.

References

[Aiken 79] Lewis R. Aiken, Psychological Testing and Assessment, Allyn and Bacon, INC., 1979

[Lance 94] Lance J. Rips J., The Psychological of Proof, Deductive reasoning in human thinking, A Bradford Book, The MIT Press, Cambridge, Massachusetts, 1994.

[MPC] Micromedium, Preparation by Computer, download demo.exe, http://www.micromedium.com

[Nowa 83] Nowakoska Maria, Quantitative Psychology: Some chosen problems and new ideas, North-Holland, Elsvier Science Publishers B.V. 1983.

[**Osiris**] Osiris Trivia Quiz. University of Sunderland (school of computing and systems), England. http://osiris.sound.ac.uk/onl-scripts/newquiz.sh

[**Robyn 75**] Robyn M. Dawes, Fundamentos y Técnicas de Medición de Actitudes, Limusa, México (1975).

[**Ronald 84**] Ronald A Berk, A guide to Criterion Referenceded Test Construction, The Jonhs Hopkins University Press, Baltimore and London 1984.

[**Ruknet**] Ruknet Cezzar, Ph.D/ Li-hsiang Ariacheo, Ph.d./ John Najariam, Ph.D/ Gary Wester Ph. D, Research and Education Association, Graduated

[**STB 97**] Software Technology Branch (STB), NASA/Lyndon B. Johnson Space Center,CLIPS release 6.04, 1997, http://www.ghgcorp.com/clips/

[FCL] Fuzzy C Language Integrated Production System, Knowledge System Laboratory, Institute for InformationTechnology, National Researc Council Canada. http://www.ghgcorp.com/clips/

[**Thorndike 14**] Thorndike, E.L., Educational Psycology, vol 1, 2, 3; New York, 1913-1914.

[Thurstone 41] Thurstone Th. G., Primary Mental Abilities of children, Educational and Psychology Means,1,1941,105-116

[**Yichie**] Yichie Shivey, MD. Massachusetts Eye and Informally, Harvard Medical School, Boston MA. DJO-Knowledge Review, Questions quiz about ophthalmology, retina quiz (may 26, '96), http://www.neii.harvard.edu/meii/kr/052696/q052696.html

[Yela 63] Yela M., Los factores de orden superior en la estructura de la inteligencia, Rev. Psicologia General y Aplicada, XVIII, 1963, 68-69

[**Zadeh 65**] Zadeh, L. A., Fuzzy Sets; Inform Contr., vol 8 (1965).

[**Zadeh 83**] Zadeh L. A. Linguistic Variables. Approximate Reasoning and Dispositions. 8th Medical Information (1983).

Probabilistic Model-Based Diagnosis

Pablo H. Ibargüengoytia[1], L. Enrique Sucar[2], and Eduardo Morales[2]

[1] Instituto de Investigaciones Eléctricas
Av. Reforma 113, Temixco, Mor., 62490, México
pibar@iie.org.mx
[2] ITESM - Campus Morelos
Paseo de la Reforma 182-A, Temixco, Mor., 62589, México
{esucar,emorales}@campus.mor.itesm.mx

Abstract. Diagnosis, in artificial intelligence, has traditionally utilized heuristic rules which in many domains are difficult to acquire. An alternative approach, *model–based diagnosis*, utilizes a model of the system and compares its predicted behavior against the actual behavior of the system for diagnosis. This paper presents a novel technique based on probabilistic models. Therefore, it is natural to include uncertainty in the model and in the measurements for diagnosis. This characteristic makes the proposed approach suitable for applications where reliable measurements are unlikely to occur or where a deterministic analytical model is difficult to obtain. The proposed approach can detect single or multiple faults through a vector of probabilities which reflects the degree of belief in the state of all the components of the system. A comparison against GDE, a classical approach for multiple fault diagnosis, is given.

1 Introduction

Diagnosis consists in determining if a particular system is functioning as it was intended; and if not, to find out an explanation of the faulty behavior, usually in terms of the part or component of the system that is not working properly.

The usual approach for diagnosis in artificial intelligence has been heuristic classification [5]. In this approach, diagnostic knowledge is represented in terms of heuristic rules that perform a mapping between measurements or symptoms and diagnostic solutions. These *shallow* representations, however, suffer from several limitations. In particular, they are usually limited to single faults and do not include all possible faults. Also, in many domains, it is easier to build a model of the system than to acquire the heuristic rules for diagnosis.

An alternative approach is *model–based diagnosis*. This consists in having a model of the correct behavior of the system. This model is compared with the observed behavior of the *real* system. From the discrepancies between the predicted and observed behavior, a diagnosis can be performed. A classical example of this approach is the General Diagnostic Engine proposed in [2]. Given a model of a device and some inputs, GDE predicts expected values and compares them against real (measured) values to identify possible faults. During the propagation process, GDE stores all the assumptions on which the predicted values rely.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 687–698, 2000.

The lists of assumptions are used to identify faulty components, but can grow exponentially in complex systems limiting its applicability. GDE was designed to work with deterministic models and in environments without uncertainty.

In this paper, a novel approach for model–based diagnosis based on a probabilistic representation of the system is proposed. The method is based on a Bayesian network model of the device and in a theory for information validation that was originally developed for sensor validation [3,4], and overcomes some of the limitations of GDE.

Bayesian networks are directed acyclic graphs (DAG) whose structure corresponds to the dependency relations of the set of variables represented in the model [7]. The nodes represent variables while the arcs represent direct dependencies between nodes. The structure of the network makes explicit the dependence and independence relations between the variables. Using the Bayesian network model, our approach detects possible faults (potential faults) in the system and its location. Then the information validation theory together with a second Bayesian network are used to identify the location of the *real* fault and eliminate apparent faults in other components. The probabilistic validation model is illustrated with a simple digital circuit, but it can be applied to more complex systems. Also, it can be used in domains in which it is difficult to have a complete deterministic model, but in which a probabilistic model can be obtained.

The next section describes the probabilistic validation model, including the theory on which the model is based.

2 Probabilistic Diagnosis Model

The probabilistic diagnosis model utilizes Bayesian networks. The diagnosis is made by the sequence of two operations: (i) *basic validation* and (ii) *isolation*. The former operation detects the presence of a fault while the later isolates the faulty element. This corresponds to the FDI (Fault Detection and Isolation) technique utilized in industry. These operations are made cyclically with all the variables in a model.

The *basic validation* consists in the estimation of the value of a variable according to the values of other related variables. A particular variable is taken as the hypothesis while the related variables act as the evidence. Thus, the propagation in the Bayesian network provides the posterior probability distribution of the variable's estimated value, given other variables. This distribution can then be used to infer if the variable has a proper value or if it shows an abnormal behavior. However, if the validation of a variable is accomplished utilizing a faulty variable then a false conclusion is expected. A basic validation algorithm then, can only tell if a variable has a potential fault, but (without considering other validations) it can not tell if the fault is real or apparent. The *fault isolation* module distinguishes between apparent and real faults, isolating the faulty variable.

The isolation module functions as follows. When a faulty variable exists, the fault will be manifested in all the related variables. The most closely related va-

riables for each variable in a Bayesian network are its *Markov blanket*. A Markov blanket (MB) is defined as the set of variables that makes a variable independent from the others. In a Bayesian network, the following three sets of neighbors are sufficient for forming a MB of a node: the set of direct predecessors, direct successors, and the direct predecessors of the successors (i.e. parents, children, and spouses) [7]. For example, consider the Bayesian model of Fig. 2 (see section 3). The MB of x consists of the set $\{a, c, f, y\}$, while MB of f consists only of $\{x, y\}$.

The set of variables that constitutes the MB of a variable can be seen as a protection of this variable against changes of variables outside the blanket. Additionally, the *extended Markov blanket* of a variable v_i written $EMB(v_i)$, is formed by its MB plus the variable itself. Utilizing these concepts, if a fault exists in one of the variables, it will be revealed in all the variables in its EMB. On the contrary, if a fault exists outside a variable's EMB, it will not affect the estimation of that variable. It can be said then, that the EMB of a variable acts as its protection against others faults, and also protects others from its own failure. The *fault isolation* module utilizes this property to update a probability of failure vector (one for each variable) and hence, to distinguish the real faulty variable [3].

Both, the detection and isolation modules utilize a Bayesian network. The detection is made using a Bayesian network representing the probabilistic relations between the variables in certain application. Figure 2 shows an example of an application with ten variables. This is called the detection network. The isolation utilizes a probabilistic causal model relating the real and apparent faults [7,8]. Fig. 3 (see section 3) shows the causal network utilized. This is called the isolation network.

The node R_i represents the real fault in variable i while A_j represents the apparent fault in variable j. The arcs denote causality, i.e., an apparent failure in variable a (node Aa) is *caused* by the existence of a real fault in one of the nodes a, c or x. This set happens to be the $EMB(a)$. Thus, given more evidence about the state of apparent faults in the variables, the posterior probability value of real faults nodes will tend either to zero or one.

The probabilistic diagnosis theory developed in [3,4] can be summarized as follows. After a cycle of basic validation of all variables is completed, a set S of apparent faulty variables is obtained. Thus, based on the comparison between S and the EMB of all variables, the following situations arise:

1. If $S = \phi$ there are no faults.
2. If S is equal to the EMB of a variable X, and there is no other EMB which is a subset of S, then there is a *single real fault* in X.
3. If S is equal to the EMB of a variable X, and there are one or more EMBs which are subsets of S, then there is a real fault in X, and possibly, real faults in the variables whose EMBs are subsets of S. In this case, there are possibly *multiple indistinguishable* real faults.
4. If S is equal to the union of several EMBs and the combination is unique, then there are *multiple distinguishable* real faults in all the variables whose EMB are in S.

5. If none of the above cases is satisfied, then there are multiple faults but they can not be distinguished. All the variables whose EMBs are subsets of S could have a real fault.

For example, consider the network shown in Fig. 2 and its corresponding probabilistic causal model shown in Fig. 3. Suppose a failure in the value of variable x. The basic validation of a will produce an apparent fault given that its estimation was made using an erroneous value. This imply the instantiation of node Aa in Fig. 3 as true. At this point, the probability of failure will show an estimation of the most suspicious real faulty nodes. Following the procedure for all nodes, will result in the detection of apparent faults in nodes a, c, x, y, and f. Thus, instantiating those apparent fault nodes in Fig. 3 will provide the posterior probability of real fault nodes R_i. This will show a high probability value in node Rx.

This probabilistic diagnosis theory and algorithm has been used in the validation of temperature sensors of a thermoelectrical power plant [3,4].

3 Probabilistic Diagnosis

This section develops the probabilistic model based diagnosis approach using the probabilistic validation model described in section 2. This will be done with the diagnosis of the electronic circuit shown in Fig. 1. $A1$, $A2$ and $A3$ represent AND gates while $O1$ and $O2$ represent OR gates.

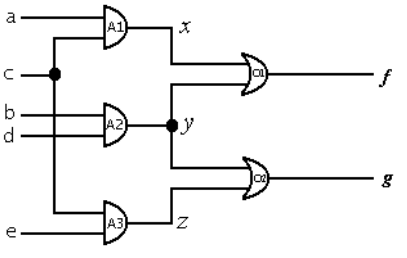


Fig. 1. Digital circuit under diagnosis.

The circuit shows three testing points that can be used if necessary: x, y, and z. Notice that verifying the correctness of variable x corresponds to the state of the $A1$ gate, and verifying f corresponds to the state of the $O1$ gate and similarly for variables y, z, and g.

This example will be used in this and the following sections to illustrate the probabilistic model based diagnosis and contrasted with diagnosis using GDE.

Figure 2 shows the detection network associated to the circuit of Fig. 1. Nodes a, b, c, d, and e represent the input variables. Nodes x, y, and z represent the variables corresponding to the operation of the AND gates and nodes f and

g represents the operation of the OR gates. Thus, the arcs are easily deduced since the probabilistic dependency corresponds to the relation between input and output variables of a device. For example, node x has a direct dependency with the value of nodes a and c. This network will be used for the detection of a fault.

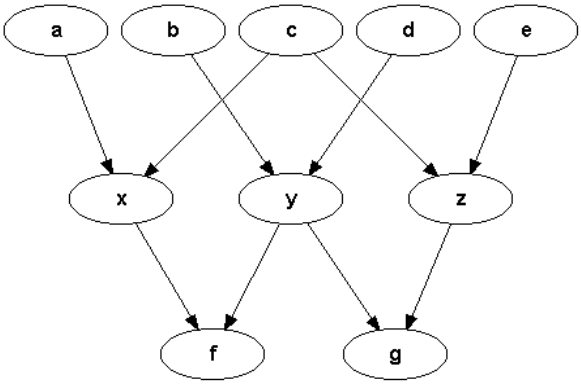


Fig. 2. Probabilistic model of the circuit of Fig. 1.

Figure 3 shows the isolation network that corresponds to the electronic circuit. Notice that the presence of arcs between the two layers (apparent and real faults) corresponds to the EMB of the nodes.

Both networks require quantitative information as part of the model. The detection network of Fig. 2 has an *a priori* probability of the root nodes (the input variables) of 50 %, i.e., a ignorance condition. Additionally, the rest of the nodes require a conditional probability matrix which reflects the *true table* of the gate. A number close to one and close to zero were used in this matrix. Regarding the isolation network of Fig. 3, the root nodes also have a 50 % probability of failure. The apparent fault nodes include a conditional probability matrix corresponding to the *noisy–or* condition [7,4].

The following experiment simulates a failure in variable f, i.e., a failure in the $O1$ gate. Consider the following inputs: $a = 1$, $b = 0$, $c = 0$, $d = 1$, and $e = 1$. The expected values for the rest of the variables is: $x = 0$, $y = 0$, $z = 0$, $f = 0$, and $g = 0$. However, consider that the failure in the $O1$ gate produces $f = 1$. Table 1 shows the steps taken by the probabilistic diagnosis algorithm and some of its partial results.

The first column indicates the variable that was just validated. The next column presents the probability of apparent fault that results from the validation. For example, on the second row, when g was validated, a 33 % of probability of detecting a fault was obtained. The third column shows the interpretation of the validation. This interpretation is important since it represents the instantiation of the apparent fault nodes of Fig. 3. The following 5 columns show the results of the propagation of probabilities in the network of Fig. 3 after the instantiation of

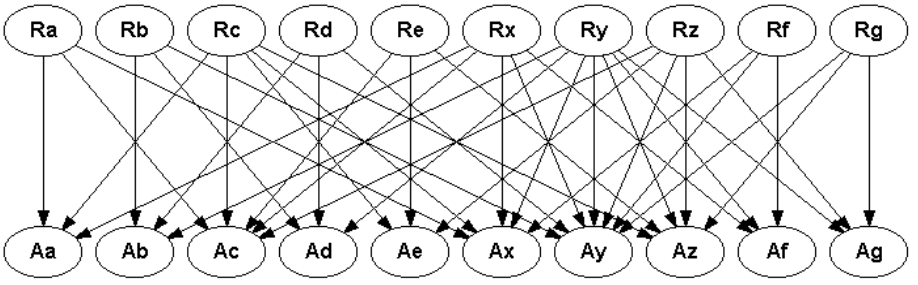


Fig. 3. Probabilistic causal model for fault isolation of the circuit of Fig. 1.

the apparent fault nodes. This corresponds to the probabilities of the real fault nodes of the figure. The last column includes a comment about the diagnosis step.

Table 1. Diagnosis experiment simulating a failure in f.

variable validated	$P(app.failure)$ (%)	state assigned	R_x	R_y	R_z	R_f	R_g	Comments
f	100	*nok*	58	58	50	58	50	without reading x, y, z
g	33	*ok*	66	12	9	66	9	without reading x, y, z
x	50	*nok*	66	12	9	66	9	reading all variables
y	0	*ok*	50	5	1	50	1	
z	0	*ok*	52	0	0	52	0	
a	50	-	52	0	0	52	0	no conclusion in detection
b	0	*ok*	52	0	0	52	0	
c	0	*ok*	10	0	0	91	0	
d	50	-	10	0	0	91	0	no conclusion in detection
e	50	-	10	0	0	91	0	no conclusion in detection

The diagnosis starts with the validation of one of the outputs namely f. The detection indicates the presence of a fault because the propagation confirms that the output must be 0 with practically 100 % of probability. Since the real value of f is 1, then a failure is detected with 100 % of probability. This produces the instantiation of Af with failure. Notice that this validation was made using only the information of the inputs and the other output, i.e., no information of points x, y and z was used. At this step, the propagation of probabilities of real faults indicates a small suspicious about a failure in x, y or f, and ignorance continues (50 %) in variables z and g. The second step validates the output g, i.e., node g of Fig. 2. The result obtained is a 33 % of probability of failure so node Ag of the isolation network (Fig. 3) is instantiated as *ok*. This second causal evidence produces a higher probability of real fault in variables x and f but it decreases

significantly for y, z and g. At this stage, looking only at the outputs f and g, there is an expected probability of real fault in f and x.

The following three steps involve the information provided by the intermediate variables x, y and z. The third step validate node x using the information of all the variables. This produces a probability of failure of 50 % which in this case can be considered as faulty. In other words, an apparent failure can be considered in x given that the information was not able to conclude an estimated value. Thus, node Ax is instantiated as nok. Notice that in this case, the probability of real failures has no change. The fourth and fifth steps validate variables y and z resulting with a total correspondence between the predicted and real value, so the nodes Ay and Az are instantiated with ok. The corresponding propagation in the isolation network produces practically no doubt about a real fault in y, z or g, but maintains a shared suspicious of a real failure in x or f.

The last five steps consist in the validation of the input variables a to e. First, the validation of a produces no results, i.e., it maintains the *a priori* consideration of ignorance in the value of the input. In this case, the interpretation consists in considering no extra information and no instantiation is made in the isolation network. Finally, the validation of b and c results in a definitive no fault in those nodes. Instantiating Ab and Ac in the isolation network results in a very clear conclusion that the fault is in node f, i.e., in the $O1$ gate.

The probabilistic diagnosis presents an additional characteristic which can be useful in applications where a timely response is required. This is, the belief measurement of the correctness of the devices provides partial results whose quality depends on the time spent in the computation. The first step shows a no significant movement of the initial ignorance condition (50 %). However, from the second step, there is a clear indication that the fault is between variables x and f, since the probability of failure of the other variables have decreased significantly. This situation changes lightly until the seventh step. In the eighth step, the conclusion is absolute since the probability of failure in y, z and g are zero, the probability of failure in x is very low but the probability of fault in f is definite.

This characteristic is important in critical applications where an approximate and fast response is preferable rather than a slow and exact diagnosis.

Consider now the experiment of simulating a fault in variable x, i.e., in the $A1$ gate. Table 2 describes all the steps.

Notice that in this experiment, the final result consists in a conclusion of a real fault in variables x and f and a strong conclusion in the correctness of variables y, z and g. According to the validation theory described in section 2, when a variable's EMB is a subset of the faulty variable's EMB, then there is no way to distinguish between a single failure in one variable and a double fault in these two variables. In this case, $EMB(x) = \{a, c, x, y, f\}$ and $EMB(f) = \{x, y, f\}$ so a simple heuristic can conclude that there is a definite fault in variable x but it is not sure about variable f.

Table 2. Diagnosis experiment simulating a failure in x.

variable validated	$P(app.failure)$ (%)	state assigned	R_x	R_y	R_z	R_f	R_g	Comments
f	100	*nok*	58	58	50	58	50	without reading x, y, z
g	33	*ok*	66	12	9	66	9	without reading x, y, z
x	50	*nok*	66	12	9	66	9	reading all variables
y	0	*ok*	50	5	1	50	1	
z	0	*ok*	52	0	0	52	0	
a	50	-	52	0	0	52	0	no conclusion in detection
b	0	*ok*	52	0	0	52	0	
c	50	-	52	0	0	52	0	
d	50	-	52	0	0	52	0	no conclusion in detection
e	50	-	52	0	0	52	0	no conclusion in detection

4 Diagnosis Using GDE

The General Diagnostic Engine (GDE) was originally proposed in [2] as an efficient general method for fault diagnosis. Its main contributions were: (i) capabilities to diagnose multiple faults, and (ii) an efficient representation based on minimal sets to diagnose in an incremental way.

GDE works under the following assumptions:

1. The act of taking a measurement has no effect on a device
2. The environment is static (the behavior of the system will not change during diagnosis)

In GDE a *symptom* is any difference between a predicted value and its measurement, A *conflict* is a set of assumptions (components that may be faulty) that supports a symptom, and thus leads to an inconsistency. A *candidate* is a hypothesis for how the actual artifact differs from the model. The goal in diagnosis is to identify, and refine, the set of candidates consistent with the observations. It explains the current set of symptoms, thus each set representing a candidate must have a non-empty intersection with every conflict.

An important idea of GDE was the use of minimal conflict sets, such that every superset of them is a valid hypothesis and there is no proper subset of any of them that could be a valid hypothesis. GDE starts with the empty set as its minimal conflict set. As the diagnosis proceeds, the candidates are increased while the conflicts are reduced.

Given a model of a device and some inputs, GDE propagates the values through the model to predict an expected behavior. Any discrepancy between the predicted behavior and the observed behavior is used to identify possible faults. GDE is based on ATMS [1] and stores, during propagation, all the components on which the predicted values rely. In the description that follows about GDE, a variable will be associated with the following information: (i) a value, (ii) a

dependency list of components that support that value, and (iii) the propagation cycle on which that value was obtained.

For instance, suppose a device called $M1$, that multiply two inputs (f and g) and produces as output f, with the following model: $h = f \times g$. Now, suppose that the value of variable $f = 4$ depends on the correct functioning of a device called $C1$ and $g = 2$ depends on $C2$, with information generated at cycle 1. This information is represented with the following notation:

$$f = 4, [C1], 1 \qquad\qquad g = 2, [C2], 1$$

GDE propagates these values to infer that $h = 8$ and that depends on the dependency lists of its inputs (i.e., $[C1]$ and $[C2]$) and on $M1$.

$$h = 8, [C1, C2, M1], 1$$

If h is measured and it differs from 8, GDE recognizes a conflict ($[C1, C2, M1]$) and is sure that at least one of these components must be faulty. This information is used, in conjunction with other conflicts, to construct minimal conflict sets. A variable can have more than one predicted value (with different assumptions of values for the variables) as shown in the example below. GDE can suffer from an exponential growth in the size of its dependency lists[1], although several extensions have been proposed to deal with this problem (e.g., [6]). ATMS propagates until:

- no more values can be propagated
- there is no sufficient information to produce a new value
- a dependency list is a superset of a conflict set

For the example of Fig. 1, GDE produces the following results.

Initial measurements:

$a = 1, [\,], 0$	$b = 0, [\,], 0$
$c = 0, [\,], 0$	$d = 1, [\,], 0$
$e = 1, [\,], 0$	

Propagation:

$x = 0, [A1], 0$	from $a = 1$ and $c = 0$
$y = 0, [A2], 0$	from $b = 0$ and $d = 1$
$z = 0, [A3], 0$	from $c = 0$ and $e = 1$
$f = 0, [A1, A2, O1], 0$	from $x = 0$ and $y = 0$
$g = 0, [A2, A3, O2], 0$	from $y = 0$ and $z = 0$

Following the same conditions presented in section 3 and first assuming a fault in $O1$, results in a measured value for $f = 1$. This measurement produces the following results: $f = 1, [\,], 1$ which produces a conflict with: $[A1, A2, O1]$ (from $f = 0, [A1, A2, O1], 0$), whose minimal conflict sets are: $[A1]$, $[A2]$, and $[O1]$. The propagation produces:

[1] For instance in a binary tree arrangement of components, the size of the dependency list at the root node (assuming that all the inputs are in the leaves) will be $2^n - 1$ where n is the depth of the tree.

$x = 1, [A2, O1], 1$ from $f = 1$ and $y = 0$
$y = 1, [A1, O1], 1$ from $f = 1$ and $x = 0$

z can not be propagated because there are no possible values for z for which $g = 0$ and $y = 1$. This is a contradiction that can appear in this type of circuits that was not previously consider by GDE [2][2]. Following with the propagation, a new value for g is obtained:

$g = 1, [A1, A3, O1, O2], 1$ from $y = 1$ and $z = 0$

Now, suppose that a measure of g with the value $g = 0$: $g = 0, [\ ], 2$. This creates a conflict of $[A1, A3, O1, O2]$ with $g = 1, [A1, A3, O1, O2], 1$. In this case $A2$ is no longer in all the conflict sets, so it is require to look for its minimal supersets with components in all the conflict sets. The minimal conflict sets are then: $[A1], [O1], [A2, A3]$, and $[A2, O2]$.

Propagating this g value produces:

$y = 0, [O2, A3], 2$ from $z = 0$ and $g = 0$
$x = 1, [O2, A3, O1], 2$ from $y = 0$ and $f = 1$
$z = 0, [O2, A2], 2$ from $y = 0$ and $g = 0$

Again, a value for z can not be evaluated (with $g = 0$ and $y = 1$).

Now suppose that measuring x gives $x = 0$ that produces $x = 0, [\ ], 3$

This creates the conflict sets: $[A2, O1]$ and $[O2, A3, O1]$, from which the only singleton minimal conflict set is $[O1]$, which strongly suggests that $O1$ is a faulty component (which is the expected result in this case).

The propagation of this value produces only $y = 1, [O1], 3$ from $x = 0$ and $f = 1$

A measurement of $y = 0$ produces $y = 0, [\ ], 4$ that creates the conflict sets: $[A1, O1]$ and $[O1]$, which again reinforces the suspicious on $O1$. Measuring $z = 0$ produces no conflicts.

As in section 3, now suppose that the failure is in $A1$ and x's value is $x = 1$. This produces $x = 1, [\ ], 3$

The new conflict set is $[A1]$ (from $x = 0, [A1], 1$), which happens to be also the only singleton minimal conflict set. The propagation of this new value encounters another aspect that was not previously considered in GDE, namely the possibility of having non-deterministic outcomes. In particular, this value of x gives two equally possible values for y:

$y = 0, [O1], 3$ from $x = 1$ and $f = 1$

and

$y = 1, [O1], 3$ from $x = 1$ and $f = 1$

In this case, a separate analysis for both cases is required (otherwise, measuring y will produce a conflict set of $[O1]$ regardless of its values, with a corresponding minimal conflict set of $[A1, O1]$, which is not the expected result).

[2] GDE supposed that if there were two values for a particular component, its third value could be established.

Analyzing both cases separately, a measure of $y = 0$ produces a conflict set of $[A1, O1]$ whose minimal conflict set is still $[A1]$. On the other hand, $y = 1$ produces conflict sets of $[A2]$ and $[O2, A3]$, with conflict sets of $[A1, A2, O2]$ and $[A1, A2, A3]$.

Continuing with the propagation, only one value of y produces a possible value for $z = 0, [O2], 3$ (from $y = 0$ and $g = 0$). In either case, measuring $z = 0$ produces no further conflicts.

Although GDE produces the expected results, in non-deterministic environments needs to branch over all the possible values, which eventually produces an exponential growth.

5 Discussion

GDE is a powerful tool for multiple fault diagnosis, it can suffer, however, from an exponential growth in the size of the dependency lists. In addition, the experiments showed two cases which were not explicitly considered in [2]: (i) what to do with inconsistent values (e.g., $y = 1$ and $g = 0$) and (ii) what to do with non deterministic models. The natural answers are, stop the simulation process (in the first case), and separately consider all the possible options, which represents a combinatorial explosion (in the second case). The exponential growth of GDE represents a serious limitation for its application to large devices or devices where a deterministic model cannot be established. On the other hand, GDE was not designed for dealing with uncertainty. In [2], a description of how to include different probabilities of failure for the components and use this information for an efficient measurement strategy is given. This mechanism, however, is independent of the diagnosis process. In particular, this information is not reflected in the propagation process or the possible values of failures that the components can take.

The proposed probabilistic diagnosis model, is suitable as GDE, for single and multiple faults diagnosis. However, contrary to GDE, it can efficiently deal with non deterministic models and include probabilities of failures into the models and into the reasoning mechanism. Its complexity grows with the size of the EMB of each variable, which in turns depends on the interconnectivity of the device to be modeled. Due to the independent assumptions of Bayesian networks, the probabilistic model could efficiently diagnose devices with a large number of components which are not heavily interconnected. In addition, the theory developed in [3,4] shows that some cases may be indistiguishable to the system. In particular, when the EMB of a variable is a subset of the EMB of a real fault, the system can be sure about the real fault but cannot tell whether another device is also faulty.

6 Conclusions and Future Work

A new approach for model–based diagnosis was presented. It is based on Bayesian networks and can deal naturally with probability of failures and with non de-

terministic models. The approach is suitable for single and multiple faults and does not suffer from a combinatorial explosion in its reasoning mechanism with larger devices (as long as they are not heavily interconnected). The probabilistic diagnosis was compared with GDE in the detection of a simulated fault in a digital circuit. The results of this comparison were discussed.

Future work is required in the integration of the information provided by the detection network, to the isolation network. At this stage, the detection module provides a binary answer corresponding to the state of the variable: *correct or faulty*. This binary value is applied to the corresponding apparent fault node of the isolation network. The proposal in the future is the direct instantiation of the apparent fault node with the continuous value (may be discretized) resulting in the propagation of the detection network.

Additionally, diverse experiments are required in order to include additional information that may enhance the model of the system. For example, the *a priori* probability of the root nodes of both networks represent now an ignorance condition (50 %). However, different devices are more immune to failure that others. Also, a mechanism that selects the best variable to validate is important in applications where a fast response is required.

References

1. J. DeKleer. An assumption based tms. *Artificial Intelligence*, 28:127–162, 1986.
2. J. DeKleer and B. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.
3. P.H. Ibargüengoytia, L.E. Sucar, and S. Vadera. A probabilistic model for sensor validation. In E. Horvitz and F. Jensen, editors, *Proc. Twelfth Conference on Uncertainty in Artificial Intelligence UAI-96*, pages 332–339, Portland, Oregon, U.S.A., 1996.
4. P.H. Ibargüengoytia, L.E. Sucar, and S. Vadera. Any time probabilistic reasoning for sensor validation. In G.F. Cooper and S. Moral, editors, *Proc. Fourteenth Conference on Uncertainty in Artificial Intelligence UAI-98*, pages 266–273, Madison, Wisconsin, U.S.A., 1998.
5. P. Jackson. *Introduction to Expert Systems*. Addison–Wesley, Wokingham, England, 1990. (2nd Edition).
6. E. Morales and H. García. A modular approach to multiple fault diagnosis. *Artificial intelligence in Process Engineering*, pages 161–187, 1990.
7. J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, Palo Alto, Calif., U.S.A., 1988.
8. Y. Peng and J.A. Reggia. A probabilistic causal model for diagnostic problem solving-parts i and ii. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17(2,3):395–406,146–162, March/April, May/June 1987.

Instance Metrics Improvement by Probabilistic Support

Héctor Jiménez[1⋆] and Guillermo Morales[2]

[1] Facultad de Ciencias Físico-Matemáticas
Universidad Autónoma de Puebla
hjimenez@fcfm.buap.mx
[2] Centro de Investigación y Estudios Avanzados del IPN
gmorales@cs.cinvestav.mx

Abstract. The use of distance functions in order to determine nearest instance class at Memory Based Learning methods may be crucial when there are no exact matchings. We add relative information over unknown feature values to improve the information extract on the training instances. An experiment was carried out for Spanish Part-Of-Speech tagging of unknown words finding a better performance with our modified function.

Keywords. Memory Based Learning, inexact matching, Spanish Part-Of-Speech tagging, unknown words.

1 Introduction

Memory Based Learning (MBL) has had many applications on natural language processing [3]. This is a powerful and very simple method, that preserves the regularities and the exceptions on the training instances with a low cost of implementation. We have chosen MBL to deal the problem of determining the Part-Of-Speech (POS) tags of Spanish words from the contexts in which they occur.

Several high quality taggers for English [1,4,10] have been proposed. Most of them have contextual information as their main support. Nevertheless, on Spanish language it is very important to consider the morphosyntactic information. The POS tagger due to Màrquez and Rodríguez [7] has meant an important advance on Spanish POS tagging. In any natural language, the problem of unknown words tagging deserves a special treatment at tagging irrestricted text. Here, we present the results of a metric used for MBL, which improves the performance of unknown words tagging.

In our POS tagger for Spanish, we use MBL to treat the tagging that requires contextual information. In general, our tagger has three steps: It tags the most frequent unambiguous words, then it tags the most frequent ambiguous

⋆ This work was supported by Mexican CoNaCyt under grant nr. 50084 and the Posgrado de Matemáticas, Facultad de Ciencias Físico-Matemáticas, UAP.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 699–705, 2000.

words and, finally, it tags the unknown words. At the first step, a dictionary with morphosyntactic information is necessary. The second step requires the morphosyntactic information and a bit more of contextual information. And the third step uses mainly contextual information, but this step is very sensitive to the unknown feature values on the test instance.

In order to tag an unknown word regarding its context, we use morphosyntactic information to determine an unknown feature value, and to achive a full distance computation. This improves the class selection whenever an exact matching fails. In an experiment, we tested the accuracy of determining POS for unknown words using the IGTree version of the MBL method and the enriched distance function.

The second section of the present paper is dedicated to analyse the metric used in MBL. In the third section, our tagger is described. An experiment carried out to compare the IGTree method with the IGTree using the new metric is presented in the fourth section. At the end, the conclusions of the present work are presented.

2 MBL and the Modified Metric

Daelemans proposed the use of MBL on natural language processing problems, and has reported important results [3]. The method can be summarized as follows: The method takes a new instance to be classified and, from a set of classified training instances, it selects the class of the nearest instance of the training set, assigning indeed this class to the new instance. The method is precise when the notion of distance between instances is defined. As an example of such distance consider the following: Given two instances $X = (x_1 \dots x_m)$ and $Y = (y_1 \dots y_m)$, let

$$\Delta : (X, Y) \mapsto \Delta(X, Y) = \sum_{i=1}^{m} \delta(x_i, y_i) \cdot w_i \tag{1}$$

where

$$\delta(a, b) = \begin{cases} 0 \text{ if } a = b \\ 1 \text{ otherwise} \end{cases} \tag{2}$$

and w_i is the *information gain* corresponding to feature i, thus, according to its importance, w_i contributes to increase the Δ value when the i-features differ. The IGTree version [2] of the MBL method compresses the training instances into a decision tree saving thus the search time and memory space. Additionaly, the MBL method faces the problem of exact matching failure –a feature value on the new instance which is not contained in any training instance– by using default information on the last matching non-terminal node. This default can be taken from the most probable class given, i.e. the most probable class for exact matchings of feature values arranged in decreasing order according to information gain. We call *maximum substring* to the string composed of these feature values.

The proposed modification to the inexact matching consists in pursuing the search even if the matching fails. In our approach, we assign a known feature value to the unknown value that occurs in the used training instances. The known feature value is supplied by *n-grams* calculated from the corpus context. Namely, let X be a new instance, Y an instance from the training base and let $\mathbf{d}om(A_i)$ be the domain of the i-attribute from the training instances. The estimation of a known feature value uses the modified distance function:

$$\delta'(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ p_i & \text{if } x_i \notin \mathbf{d}om(A_i) \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

where $p_i = 1 - \Pr(c\ y_i|c)$, and c is a context of x_i. Strictly speaking, the resulting function Δ is not a metric since it lacks the "symmetric property" of metrics (if a pair is conmuted Δ may be undefined), nevertheless it can be extended to a metric. With this in mind we will call Δ a metric as well. Certainly, we use *contexts* around each word to be tagged, moreover we use contexts around the unknown feature values to estimate known feature values i.e. calculating the probability of y_i occurring in the context of x_i, see figure 1.

training instance	$y_1 \ldots$	$\ldots y_{i-1}\ y_i\ y_{i+1} \ldots$	y_m
new instance with an unknown feature value	$x_1 \ldots$	$\ldots x_{i-1}\ x_i\ x_{i+1} \ldots$	x_m
context of x_i in the new instance		$x_{i-k} \ldots x_{i-1}\ x_i\ x_{i+1} \ldots x_{i+k}$	

Fig. 1. Context of an unknown feature value taken from the new instance.

The use of p_i resembles the use of information gain. If the feature value y_i is the most probable one to be in the new instance context, then this value will provide a low weight for the distance computation. This new weight will modify the distance value according to the probability-complement of finding a known feature value in the new instance.

Now, the IGTree is modified making use of δ' as follows: Given a new instance X and a training base,

1. If any of feature value of X with the highest weights is unknown by the training instances, then we apply δ' in order to select the nearest instance class.
2. Otherwise, we continue the IGTree computation.

Due to the fact that $\Pr(c\ y_i|c)$ has an inverse relation with δ', the first step of the above procedure chooses the class of that instance which contains the feature value y_i with maximum probability, because this y_i is such that $\Pr(c\ y_i|c)$ minimizes the distance.

Table 1 shows a very simple example of the trigrams probability, in where `SUST`, `ADV`, `NUM`, `PRON`, `VERB`, `NP` and `CONJ`, stand for noun, adverb, numeric value, pronoun, verb, proper noun and conjunction, respectively. Suppose that:

- a new instance, X, has one of the highest weights attribute, say the second attribute,
- the second feature value in X is `ADV` and
- `ADV` $\notin \mathbf{d}om(A_2)$.

In this case, the IGTree method uses the most probable class to assign the corresponding POS tag to X, whereas the modified IGTree method with δ' makes the matchings with all instances as if the second attribute would be known but weighting it now with p_2. Thus, if

$$Y = (\texttt{dor SUST del CONT})$$

is a training instance to tag an unknown word, and the new instance is:

$$X = (\texttt{dor ADV , COM})$$

with the unknown second feature value context :

$$c = (\ldots \texttt{de dor , } \ldots),$$

then we use table 1 because we have supposed `ADV` $\notin \mathbf{d}om(A_2)$, and we obtain $\delta'(x_2, y_2) = 0.333333$. Of course, this process is possible on taking into account the trigram information: whenever a feature value is not in the corresponding attribute domain we try to use the trigram probability according to the feature value context to compute the distance value.

Table 1. Conditional probability to find some feature value between two word endings.

end_{-1}	end_1	x	$\Pr(end_{-1}\ x\ end_1 \mid end_{-1}\ end_1)$
de	,	SUST	0.666667
de	,	ADV	0.025641
de	,	NUM	0.0512821
de	,	PRON	0.025641
de	,	VERB	0.0512821
de	,	NP	0.128205
de	,	CONJ	0.0512821

3 The Tagging Process

We have begun the task of building a POS tagger with very limited linguistic resources. A corpus[1] of 10,000 tagged words has been used at this first stage of development. The tagging process was divided on three parts according to the

[1] This is a part of *Corpus del Español Mexicano Contemporáneo* [6], kindly supplied by El Colegio de México.

types of words: defined frequent words (words having just one tag in any context), ambiguous frequent words (words having at least two different contexts with different POS) and unknown words (all remaining words). The tagger should accomplish the following tasks:

1. To tag the defined words.
2. To disambiguate ambiguous words.
3. To determine the POS of unknown words.

At each step, the tagger uses a special type of resource. The defined words tagging is based on a dictionary and morphosyntactic rules. The same is used for ambiguous words tagging plus the context of these words. For unknown words, the major support is provided by the context, only a portion of morphosyntactic information is still used.

The involved resources are the dictionary, the list of morphosyntactic rules and the extracted contexts from the corpus. The dictionary is structured by sections: defined frequent words, ambiguous frequent words, very frequent verbal forms, punctuation signs and proper nouns. The criteria for different word types inclusion are determined by the analysis of the frequent words[2].

The list of morphosyntactic rules includes the rules based on the conditional probability of trigrams, verb conjugation [9] rules for regular and irregular verbs, and rules to determine the POS of a word [8] from a suffix.

The contexts used were formed by tags and endings of the neighboring words of the corpus. The MBL method was applied to three different problems: to disambiguate ambiguous words, to disambiguate the verb/nominal ambiguity, and to determine the corresponding POS to unknown words. Using around 9,000 words, we obtained an average near to 94% on accuracy.

4 The Experiment

After the step of disambiguating the verb/nominal ambiguity, two tests were carried out to tag unknown words: the first one applying the IGTree method and, the second one applying the IGTree method with δ'.

In order to define the context c (referred in eq. 3 by p_i) of an unknown feature value contained in a new instance, an ending word of left and right contiguous words for the words whose feature value is unknown was used.

Each of the two tests was carried out by dividing the corpus on ten parts. One of them is chosen as the test text. The training text is successively increased with the other nine parts. Once the unknown words are tagged the accuracy, of the IGTree and IGTree with δ', is determining which is shown at Fig. 2.

Indeed, the probability p_i is better determined as the corpus grows. Therefore, the estimation of an unknown feature value will have higher accuracy with

[2] Our treatment is based on frequent word and considers the one hundred most frequent words. Thus, the result provided by Guiraud [5] holds: the words with ranked up to position 100 cover 60% of the lexicon.

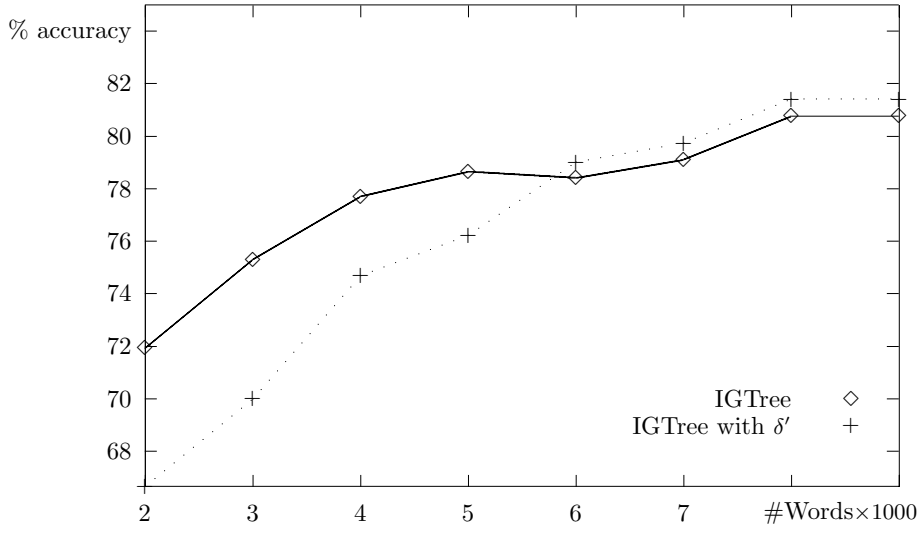


Fig. 2. Performance of IGTree and the use of δ' on unknown words tagging.

a larger corpus because it increases the number of contexts for each unknown feature value, and the computation of the probability is thus better precised. Also, as the corpus grows, the behaviour of the IGTree method, from the point of view of the maximum substring introduced at the begining of section 2, adds new substrings and precises the IGTree information gain. Nevertheless, no information on unknown features values like probability of trigrams is used.

The performance result in the experiment carried out shows that after 5,000 words of training corpus, the δ' modification of IGTree has a slightly higher accuracy than the IGTree pure method. It is necesary to carry out a new experiment with a larger corpus to determine a definitive significance of the δ'-IGTree on the IGTree method. Màrquez and Rodríguez show some performance data on unknown word problem, in POS tagging: their decision tree-based method reaches 83.3% on accuracy and the IGTree method 79.18%. But it is uncomparable with the δ'-IGTree method because this experiment was carried out under different conditions (e.g. tag-set and corpus).

5 Conclusions and Future Work

POS tagging of unknown words is a very important problem to solve in natural language processing. Also, it is necessary to offer a solution to unknown feature value problem, because learning procedures frequently faces this problem.

A modified metric to deal the problem of non exact matching into the MBL method has been presented. This metric was tested on unknown words POS tagging, using a POS tagger on development.

At the present time, we have an initial version of a POS tagger, still unpractical for real problems. The next step in this direction is to grow the corpus to 50,000 words. In spite of the fact that the corpus is very small, the present results allow us to explore about new metrics for the MBL method. The reason why the proposed metric improves the implicit metric on the IGTree method is due to the fact that the new metric is enriched with additional information.

In any particular case where some unknown feature value is met, the new metric requires a full computation of the distance, and this is clearly contrary to the more natural procedure of picking up one of the most probable classes from the IGTree method.

In the present application, the attributes to deal with an unknown feature value of any new instance were taken from the context in which the unknown feature value appears. This is a particular case. In general, research should be carried out in order to select the attributes depending on each purpose. Also, in order to reinforce the metric, a method should be used with a better efficiency with respect to the method offered by the probability of *n-grams*.

References

1. Eric Brill: Some advances in transformational-based part of speech tagging. In *Proceedings of 12th National Conference on Artificial Intelligence*, pp 722-727, 1994.
2. Walter Daelemans, Antal van den Bosch, Ton Weijters: IGTree: Using Trees for Compression and Classification in Lazy Learning Algorithms. In D. Aha (ed.) *Artificial Intelligence Review*, special issue on Lazy Learning, 1996.
3. W. Daelemans, A. van den Bosch, J. Zavrel, J. Veenstra S. Buchholz & B. Busser: Rapid development of NLP modules with memory-based learning. In *Proceedings of ELSNET in Wonderland*, Utrecht: ELSNET, pp 105-113 1998.
4. A. Franz: *Automatic ambiguity resolution in natural language processing, an empirical approach*, Springer Verlag, 1996.
5. Pierre Guiraud: *Les caractères statistiques du vucabulaire*, Mouton, 1954.
6. L.F. Lara, R. Ham Ch. & M.I. García: *Investigaciones lingüísticas en lexicografía*, Jornadas 89, El Colegio de México, 1979.
7. Lluís Màrquez and Horacio Rodríguez: Part-Of-Speech tagging using decision trees. In *Lecture Notes in Artificial Intelligence No. 1398*. ECML-98, Springer Verlag, pp 25-36, 1998.
8. José G. Moreno De Alba: *Morfología derivativa nominal en el español de México*. Ed. UNAM, México, 1986.
9. Santiago Rodríguez and Jesús Carretero: A Formal approach to Spanish morphology: the COES tools. From `http://www.datsi.fi.upm.es/ coes`, 1997.
10. R. Weischedel, M. Meteer, R. Schwartz, L. Ramshaw and J. Palmucci: Coping with ambiguity and unknown words through probablistic models. In *Computational Linguistics*, 1993.
11. Jakub Zavrel and Walter Daelemans: Recent Advances in Memory-Based Part-of-Speech Tagging. In *VI Simposio Internacional de Comunicación Social*, Santiago de Cuba, pp. 590-597, 1999.

EDAS – Event-Disturbance Analysis System for Fossil Power Plants Operation

G. Arroyo-Figueroa[1] and L. Enrique Sucar[2]

[1]Instituto de Investigaciones Eléctricas
AP 475-1, 62001 Cuernavaca, Morelos, Mexico
garroyo@iie.org.mx
[2]ITESM Campus Morelos
AP C-99, 62020 Cuernavaca, Morelos, Mexico
esucar@campus.mor.itesm.mx

Abstract. A methodology for on-line diagnosis and prediction of power plant disturbances has been developed, implemented, and tested. The approach is sufficiently comprehensive to enable a wide variety of disturbances to be analyzed correctly and efficiently. The analysis is based on a novel knowledge representation, called Temporal Nodes Bayesian Networks (TNBN), a type of probabilistic network that include temporal information. A TNBN has a set of temporal nodes that represent state changes. Each temporal node is defined by a event and a time interval associated to its occurrence. The method has been implemented and integrated with a power plant training simulator. Disturbance models for the feedwater and superheater systems have been developed and implemented as the knowledge database for the disturbance analysis system.

1 Introduction

The on-line analysis systems currently used in power plants are designed to monitor and analyze limited aspects of plant operation [1]. The interpretation of the results for the overall systems must be evaluated by the operator. Such interpretation may be easy in principle, but in practice it is prone to errors in operator judgment due to the large number of variables and the infrequent occurrence of the disturbances.

Computer systems that can be used for logging the sequence of events are commonly used. A resolution of a few milliseconds is typically available and a large number of events can be recorded. Basic functions, such filtering, consistency checking and checking for reasonability, are generally performed; however, only limited analysis (information integration) is available. These systems have proven helpful in post-trip analysis, but the give very little assistance to the operation during the progress of disturbances.

A conventional annunciator system activates an alarm on the basis of a single variable being in an off-normal condition. A message based on two or more conditions is conceptually a slight extension of using one condition only. In practice, however, the conventional annunciator systems include only very limited capabilities of this kind. The most effective means of implementing these simple features on a digital computer is to use a straightforward table lookup method. However, the analysis becomes more intricate when it is necessary to recognize the sequence of

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 706—717, 2000.

events to diagnose the underlying cause [2,3]. A further complication arises when it is desired to keep the operator informed about the progress of the disturbance. In this case, a cause-consequence analysis is required [4,5,6]. The main characteristic of a cause-consequence analysis should be modeling the propagation of events and a mechanism for updating messages as the disturbances propagate.

Consider a power plant where a number of digital signals can be monitored. A signal exceeding its specified limits (high or low) is said to be an event. Events that have the same underlying cause are considered as one disturbance. If the events are not diagnosed and rectified in a timely manner, the situation may lead to a deteriorating plant condition and eventually initiate the actuation of plant protection system, which will shut down the plant. In this paper we propose a novel methodology for modeling the propagating of events in power plants. The method used for this analysis is called "temporal nodes bayesian network", an extension of propabilistic networks [7]. A TNBN has a set of temporal nodes that represent state changes. Each temporal node is defined by a event and a time interval associate to its occurrence. A demonstration-type Event Disturbance Analysis System (EDAS) has been developed as result of application of this methodology. The development of the EDAS is motivated by the desire to improve plant availability through early diagnosis of disturbances that could lead to plant shutdown.

The paper is organized as follows. Section 2 describes the architecture of the Event Disturbance Analysis System. In section 3 the cause-consequence methodology based on temporal node bayesian networks is presented. Section 4 illustrates its application in a fossil power plant with a detailed example; the event disturbance analysis of the feedwater and superheater systems. Finally, section 5 presents conclusions and future work

2 Description of EDAS

2.1 Overview

Figure 1 shows the architecture of EDAS [8]. The data acquisition module receives the signals in the form needed for the event disturbance analysis. The event detection compares the measured values with their high and low limits. An event is said to have occurred when a variable that previously was within its limits goes outside its limits. A log is maintained of the time when the limit violations occur. The EDAS analyzer is activated when one or more events have been registered. The messages resulting from the EDAS analyzer are automatically displayed and kept up-to-date.

2.2 Data Acquisition System

The data acquisition system is the interface between the instrumentation and EDAS. This system samples the input signals, checks for consistency (whenever multiple sensor readings are available) and sensor validity, and converts the signals to engineering units. The data acquisition system is designed as a separate module to enable existing power plant data acquisition capabilities to be utilized with EDAS.

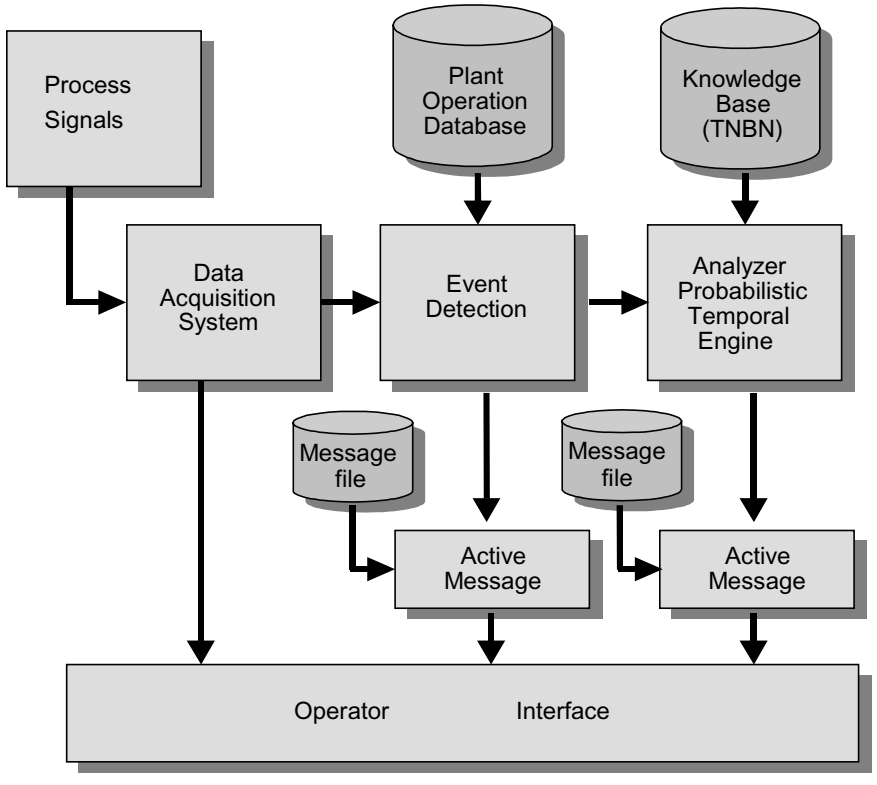


Fig. 1. Simplified Diagram of the architecture of EDAS

2.3 Event Detection

The EDAS event detection activates and schedules the execution of the disturbance analysis. This module first communicates with the analyzer to determine the latest status of each process variable. It updates user-defined derived variables and variables limits. Then, the event detection module compares the variables with their specified high and low limits to determine if any events have occurred since the last sampling. The module gives the analyzer a copy of relevant data.

2.4 Analyzer

The disturbance analysis is performed separately from other modules, so that data acquisition can continue while the disturbance analysis is under way. The main functions of the disturbance analyzer are (a) propagate the evidence; (b) obtain the probability of the main disturbance; and (c) obtain the probabilities of the past and future events. A complete description of the cause-consequence analysis is presented in the next section.

2.5 Operator Interface

The operator interface presents the process signal and translates the activated message identifiers into text and graphic displays, such that these are accesible to the operators that can be displayed on an interface. The operator interface system is essentially independent of the other EDAS modules, thus facilitating modifications or entire redevelopment of this module.

2.6 Data Base

The databases have been divided into broad classifications: (a) the dynamic data base, (b) the model database and (c) message database. The dynamic data base contains the most important attributes associated with each plant variable: name and value, high and low limits, dead-band, priority, time event, and status indicators. It is updated each sampling interval by the data acquisition system The model database contains the information that represents the disturbance models of the plant. These disturbance models are temporal node bayesian networks. The message database contains a list of messages. The list of messages gives the information for each possible event and the results of the disturbance analysis.

3 Cause-Consequence Analysis

A cause-consequence analysis is performed based on a temporal node bayesian network (TNBN), which is used to model the temporal and causal relationships between variables. Our advisory system employs Bayesian networks as a high-level reasoning tool for incorporating inherent uncertainty for use in probabilistic inference. Due to the large amount of experience available with bayesian networks [9], it is natural to try to use modifications of this technique for including temporal relation in on-line analysis [10,11,12].

3.1 TNBN Definition

A Temporal Network Bayesian Network (TNBN) is a formal representation of causal and temporal relationships between events. In other words, a TNBN is a model of system or plant behavior disturbances that can be compared to actual plant response. A TNBN is defined as follows [13]:

> **Definition 1.** A Temporal Nodes Bayesian Network is defined as **TNBN=(V, E)**, where **V** is the set of temporal nodes and **E** is the set of edge. Each temporal node is defined by an ordered pair (σ, τ) and the conditional probability matrix that specifies the probability of each ordered pair given its parents.

The values of each TN can be seen as the "cross product" between the set of values (Σ) and the set of time intervals (T), except for the default state, which is associated to only one interval. The definition of the TNs for the feedwater and superheater systems

is presented in fig. 2. TNs are connected by edges. Each edge represents a causal-temporal relationship between TNs. The conditional probability distribution for each node is defined as the probability of each ordered pair (σ_i, τ_i) given the ordered pairs of its parents (σ_j, τ_j).

In each temporal node, the temporal intervals are relative to the parent nodes; that is, there is not an absolute temporal reference. This makes the representation more general; but, for its application, we need to associate these relative times to the actual or absolute times of the observed events. We developed a mechanism for transforming the relative times to absolute, based on the timing of the observations. In the next section we present the inference mechanism.

3.2 Disturbance Analysis Methodology

The cause-consequence analysis is used to indicate that the primary emphasis is on determining causes of events that have occurred, as well as predicting future consequences. The rule based approach typically used for conventional expert systems is abandoned in this work. This is because of the inability of rule-based approaches to property model the inherent uncertainties and complexities of the relationships involved in the diagnosis and prediction of disturbances. The TNBN formalism includes the dealing of uncertainty and time for the diagnosis of disturbances and prediction of events, using the detector readings as important pieces of information. The analysis stars when an event is detected by the event detection module. This module takes the readings of the sensor and compares its value with the low and high limits. A signal exceeding its specified limits is said to be an event.

The inference mechanism of a TNBN model is based on the detection of events and the propagation of the evidence. The inference mechanism updates the marginal posterior probabilities of each node (variable) of the network given the occurrence of an event or events [9]. We define ***tc*** as the time when an event is detected and α as the *real time occurrence function*, this function is define as the absolute value of the difference between the time of occurrence of a pair of connected events. As the net does not have any temporal reference, the time of occurrence of the first event fixes temporally the network. The value of α is used to determinate the time interval of the "effect" node considering the "cause" node as initial event. Afterwards, the evidence is propagated through the network to update the probabilities of the other nodes. These probabilities show the potentially occurrence of the past and future events. The stop condition is when a terminal or leaf node is reached.

We can identify three main steps in the inference mechanism procedure:

1. detection of the event or events occurrence and definition of the time interval of occurrence of the event(s);
2. propagation of the evidence occurrence through the net and update of the probabilities of the variables;
3. determination of the potential past and future events.

Step 1. Event detection and time interval definition

When an initial event is detected, its time of occurrence, "$\mathbf{tc}_{\mathbf{initial}}$", is utilized as temporal reference for the network. There are two possible cases, depending on the

position of the *initial* node in the network: (a) the initial event corresponds to a root node, (b) the initial event corresponds to an intermediate or leaf node.

Step 2. Propagation of the evidences
Once the value of a node is obtained (time interval and associated state), the next step is to propagate the effect of this value through the network to update the probabilities of other temporal nodes.

Step 3. Determination of the past and future events
With the posterior probabilities, we can estimate the potentially past and future events based on the probability distribution of each temporal node. If there is not enough information, for instance there is only one observed event that corresponds to an intermediate node, the mechanism handles different *scenarios*. The node is instantiated to all the intervals corresponding to the observed state, and the posterior probabilities of the other nodes are obtained for each scenario. These scenarios could be used by an operator or a higher level system as a set of possible alternatives, which will be reduced when another event occurs (see Arroyo, 1999 [13] for details).

3.3 Example of the TNBN

This section shows four possible disturbances in two systems of a power plant: the feedwater system and the superheater steam system. The four disturbances are a power load increase (LI); a feedwater pump failure (FWPF); a feedwater valve failure; and the spray water valve failure (SWVF). The feedwater system is designed to maintain sufficient water inventory in steam generators during power generation by supplying the required feedwater flow to the generator. Figure 2 shows a simplified diagram of the power plant's systems. The system encompasses the piping and associated instrumentation from the inlet of the feedwater pumps to the steam generators. The flow path includes the steam driven feedwater pumps, the high-pressure heaters and the feedwater control valves (main and by-pass). The flow is controlled by a combination of valve position and feedwater pump speed adjustment. The superheater steam system is designed to regulate the flow and temperature of the superheater steam to high-pressure turbine. The system encompasses the spray water valve and associated instrumentation from the inlet of the steam flow to turbine. The flow path includes spray valve, steam valve, and superheater exchange. The steam valve controls the steam flow. The temperature is regulated by the attemperation flow of the spray water valve and by the slope of burners. The operator's task includes a complex information-filtering and integration function in order to control properly the feedwater and superheater systems. Furthermore, a fast (often one-minute o less) response is needed to prevent a plant trip once an alarm is activated

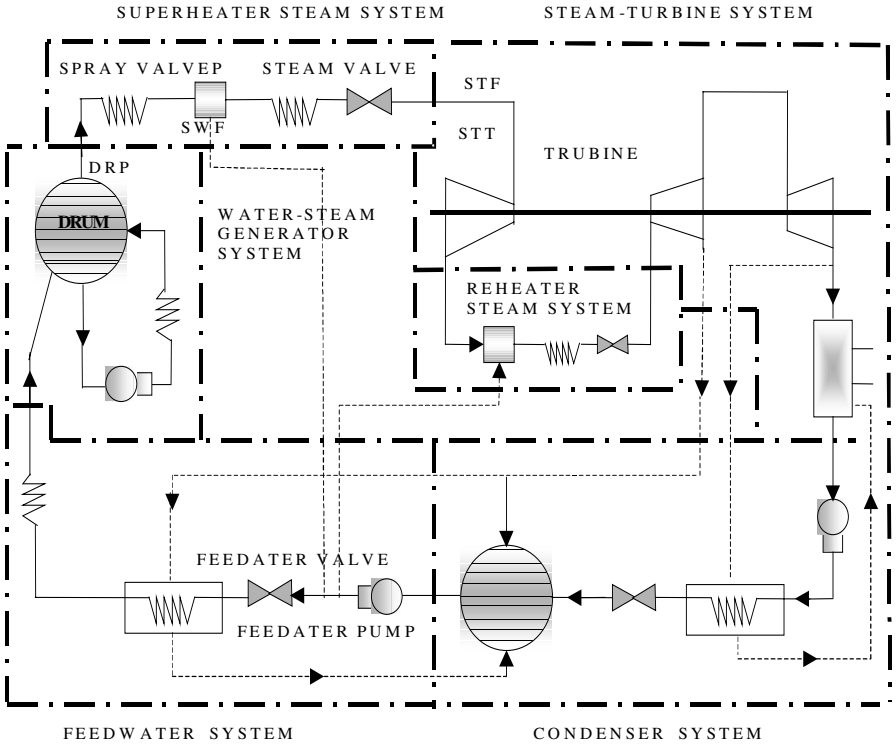


Fig. 2. Simplified diagram of fossil power plant

Figure 3 shows the temporal node bayesian network that represents the events sequence for the four disturbances in the feedwater system and steam superheater system. The four disturbances: LI, FWPF, FWVF, SWVF, appear as root nodes, these nodes do not have time intervals. Each disturbance produces an event sequence. For example an increase in the power load demand generates a steam valve increase, a steam flow increase, a drum level decrement, and a steam temperature decrement; but also generates a feedwater current pump increase, a feedwater flow increase and drum level increase. To determinate which of the disturbances is present it is a complicated task, because there are similar sequence of events for the four main disturbances. We need additional information in order to determine which is the real cause. In particular, the temporal information about the occurrence of each event is important for an accurate diagnosis. For example, a feedwater flow increase (FWF) can be caused by two different events: the feedwater pump current augmentation (FWP) and feedwater valve opening increase (FWV). We can use the time difference between the occurrence of each event, FWV-FWF and FWP-FWF, for selecting the "cause" of the increase of the FW flow. According with the process data, the time interval between a pump current augmentation and an increase of the flow (FWP-FWF) is from 25 to 114 seconds. The time interval between the valve opening increase and an increase of the flow (FWV-FWF) is from 114 to 248 seconds. Hence, if the flow increase occurs in the first time interval, the probable cause is an augmentation of the pump; but if the flow increase occurs in the second time interval, the probable cause is a valve opening increase

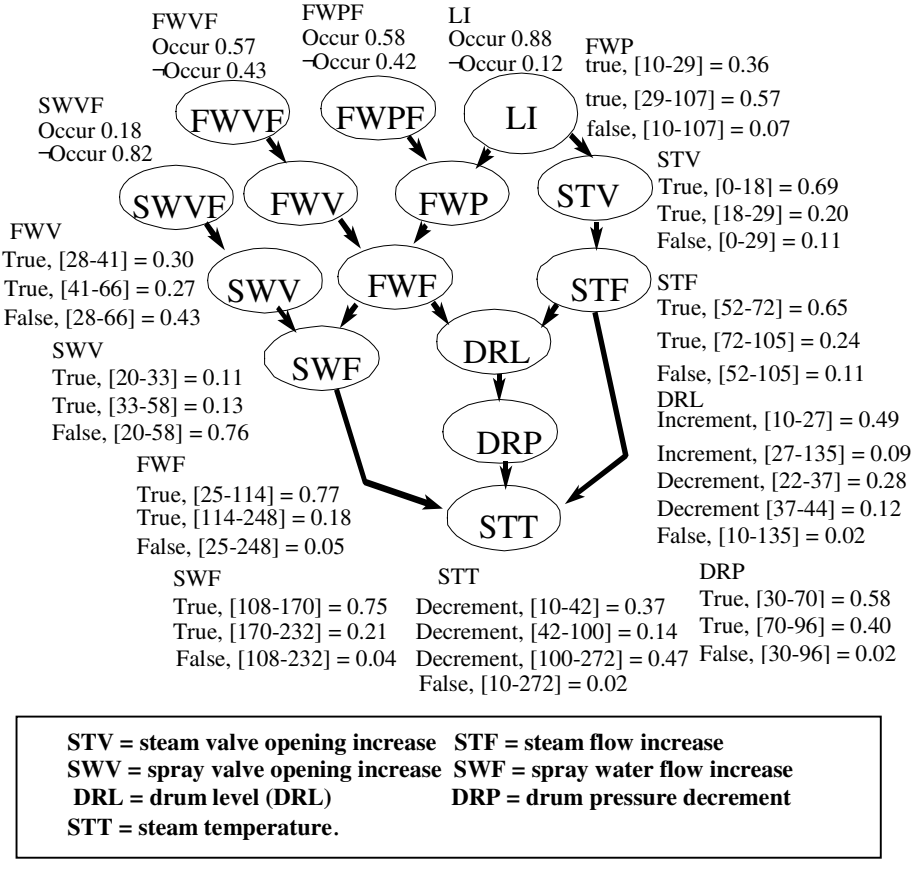


Fig. 3. TNBN for the steam superheater and feedwater systems.

The network structure was defined based on knowledge of an expert operator. The definition of the time intervals and the probability parameters for each temporal node was obtained based on knowledge about the process dynamics combined with data from failure simulation test of a power plant simulator.

4 Results

4.1. Performance Evaluation

EDAS was evaluated testing four disturbances in a fossil power plant training simulator. The analysis was performed as follows. The selected disturbances were first simulated on the simulator. During these simulations, the EDAS was actively performing its analysis, and relevant information was sent to the acquisition system. For all the modeled failures, the general arrangement of the messages is as follows:

- The first message contains a description of the particular event that was detected. It shows the plant status through the relevant variables, indicating for each one its value, status and tendency.
- The second message is activated by operator request or when the steam generator has reached a critical alarm condition. This message informs the probable disturbance, the suggested recovery and the predicted consequences.

The analysis starts when a *feedwater flow increase* (**FWF**) is detected at 4:38:00. This time is called the occurrence time of the event ($\mathbf{tc}_{\mathbf{FWF}}$). The related variables are *feedwater valve* (**FWV**), *feedwater pump* (**FWO**) *and drum level condition* (**DHL**). The status of these variables is "normal" only with a positive increase for the case of **FWP**. Figure 4 shows the relevant information about de detected event.

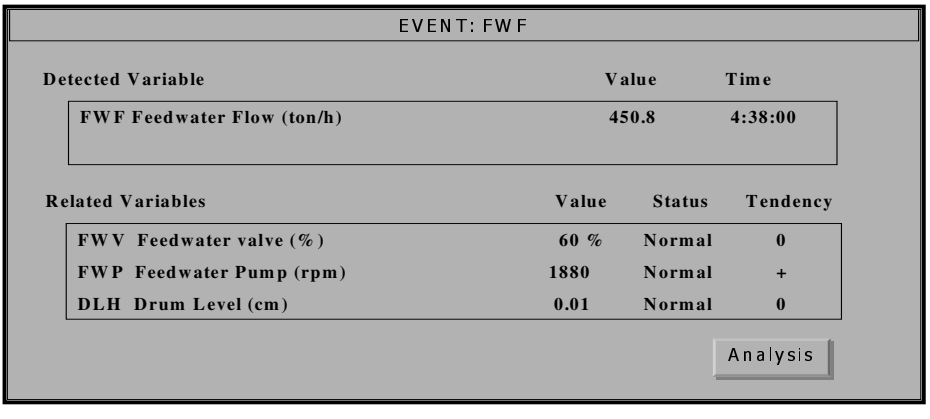


Fig. 4. Information about the event detected and its related variables.

Figure 5 shows the probability value of the initial event and the probabilities of the past and future events and its time interval of occurrence. If the FW flow occurs in the first time interval, the most probable disturbance is the FW pump failure (the velocity of the pump tends to maximum) with a probability value (certainty) of 0.75. The recommended actions are (1) run the stand bye FW pump and (2) repair the FW pump. The recommended actions are in the Plant Operation Database and they are update by the analyzer system. The probable past events are a FW pump increase, 25 to 114 seconds before the FW flow event, with a probability of 0.61; and a FW valve increase, 114-248 seconds before the FW flow event, with a probability of 0.32. The probable future event is a high level condition 10 to 27 seconds after the FW flow event detection with a probability of 0.95.

Figure 6 shows the conclusions when a steam temperature decrease occurs. If the steam temperature decreases, the most probable cause is a power load increase with a probability of 0.92. The recommended actions are (1) check the steam flow and (2) check the water flow spray. The probable past events are a steam flow increase, 10 to 42 seconds before the steam temperature decrease, with a probability of 0.96; a drum pressure decrease, 100 to 272 seconds before the steam temperature decrease, with a probability of 0.62; and spray water flow increase, 42-100 seconds before the steam temperature decrease, with a probability of 0.36. There is no probable future event because it is a terminal node.

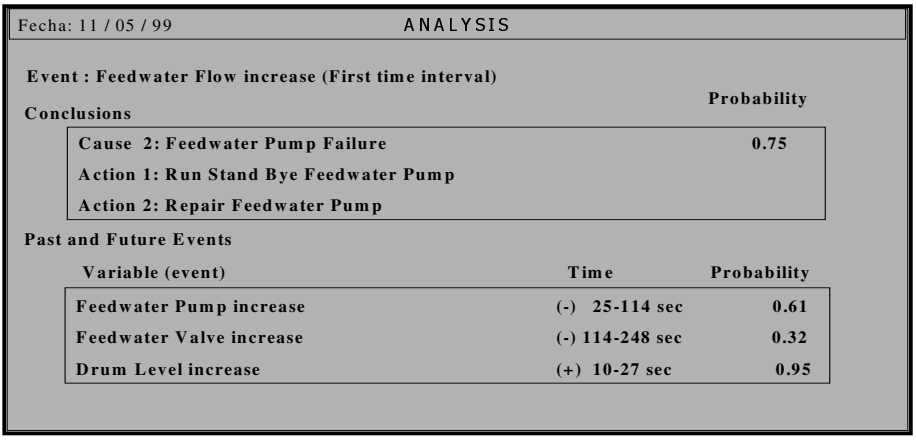


Fig. 5. Diagnosis and prediction for the FWF occurrence

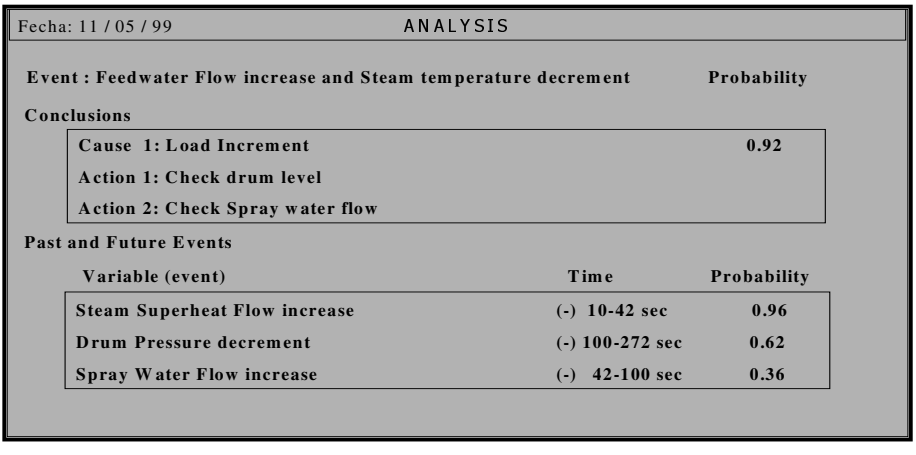


Fig. 6. Diagnosis and prediction for the STT occurrence

These results show the prediction and diagnosis capabilities of the TNBN model in a real process. SEDRET provides the operators with an early and precise diagnosis for disturbances. We are encouraged by the fact that the model can produce a reasonable accuracy in times that are compatible with real time decision making. This is important for situations in which the operator must take the best control action to avoid a shutdown of the power plant.

4.2 Experimental Results

Table 1 summarizes the results of simulating failures for the four disturbances of the feedwater and superheater systems. The process data was generated by a full scale simulator of a fossil power plant. We selected 80% of this data-base (800 registers) for parameter learning and 20% (200 registers) for evaluation. The model was evaluated empirically using two scores: accuracy and a measure based on the Brier

score (total square error). The Brier score is defined as: $BS = \Sigma^{n}_{i=1} (1 - P_i)^2$. P_i is the marginal posterior probability of the correct value of each node given the evidence. The maximum Brier score is: $BS_{MAX} = \Sigma^{n} (1)^2$. A relative Brier score is defined as: RBS (in %) = $\{1 - (BS / BS_{MAX})\} \times 100$

Table 1. Empirical evaluation results

Parameter	μ	σ
Prediction		
% of RBS	87.37	9.19
% of Accuracy	84.48	14.98
Diagnosis		
% of RBS	84.25	8.09
% of Accuracy	80.00.	11.85
Diagnosis and Prediction		
% of RBS	95.85	4.71
% of Accuracy	94.92	. 8.59

The results of the evaluation are shown in terms of the mean and the standard deviation for both scores. These results show the prediction and diagnosis capacity of the temporal model in a real process. Both scores are between 80 and 97% for all the set of tests, with better results when intermediate nodes are observed, and slightly better results for prediction compared to diagnosis. We consider that these differences have to do with the "distance" between assigned and unknown nodes and with the way that the temporal intervals were defined.

5 Conclusions

This paper presented the design and implementation of an event and disturbance analysis system, called as EDAS. EDAS has three main tasks: (1) detection of events; (2) analysis of cause-consequence of the events; and (3) prediction of future event and its time interval of occurrence. This system is used to assist an operator in real-time assessment of plant disturbances and in this way contribute to the safe and economic operation of power plants. Development of EDAS was initially motivated by the desire to improve plant availability through early diagnosis and prediction of events and disturbances that could lead to plant shutdown. An intelligent framework for event and disturbance analysis has been a primary objective throughout the project. The framework is based on a novel methodology for dealing with uncertainty and time called TNBN. The inference mechanism consists in the propagation of evidences through the net using a probabilistic inference mechanism. When an event is detected, the inference mechanism updates the marginal probabilities of each node (variable) of the network. This probabilities are used to determinate the most probable disturbance and the most probable event occurrence. TNBN model is based on representing changes of state in each node. If the number of possible state changes for each variable in the temporal range is small, the resulting model is much simpler. This

facilitates temporal knowledge acquisition and allows efficient inference using probability propagation. At present, the probabilistic intelligent system is just a prototype to be tested in a real fossil power plant. Our future work will be focused on developing and integrating our approach in a distributed control system for power plants. Also, we are interested in developing an automatic learning mechanism of the network structure and time intervals based on process data, that can facilitate the construction of the probabilistic models.

References

1. Arroyo-Figueroa G., Villavicencio A.: Intelligent System to Improve heat rate in fossil power plants. In Proceedings of IFAC Symposium on Artificial Intelligence in Real-Time Control, Valencia Spain, (1994), 365-371.
2. Falinower C., Mari B.: SACSO: An Expert System for Fossil-Fuel Power Plant Operations. Technical Report HI21/94-006, Electricité de France, Paris (1994).
3. Tsou J.: HEATXPRT Heat Rate Improvement Advisor Prototype Version. User Manual Electric Power Research Institute, Palo Alto California, (1995)
4. Meijer C.H., Frogner B.: On-Line Power Plant Alarm and Disturbance System. Final Report NP-1379 Electric Research Institute, Palo Alto California, (1980).
5. Wong A. C., Ho H., Teo C.: Use of an expert system shell to develop a power plant simulator for monitoring and fault diagnosis. Electric Power System Research, (1994), **29**, 27-33.
6. Shirley S, Forbes H., Nelson R.: An on-line expert system to improve heat rate. In *Proceedings of the 1990 American Power Conference*, Miami Fl, (1990), 314-321.
7. Arroyo-Figueroa G., Sucar E.: Temporal Bayesian Network for diagnosis and prediction. In Laskey K., Prade H.: Proc. 15th Conference on UAI, (1999), 13-20.
8. Alvarez-Salas Y.: Diagnosis Expert System based on Temporal Bayesian Network for level drum control of steam generator of fossil power plant (In Spanish), B.S. Thesis, Universidad de Occidente, (1999).
9. Pearl J.: Probilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Mateo, CA. (1988).
10. Aliferis C.F., Cooper G.: A Structurally and Temporally Extended Bayesian Belief Network Model: Definitions, Properties, and Modeling Techniques. In Procc. of 12th Conference on UAI, (1996), 28-39.
11. Santos Jr. E., Young J.: Probabilistic Temporal Networks. Report AFIT/EN/TR96-006 AFIT, (1996).
12. Kang CW, Golay MW.: A Bayesian belief network-based advisory system for operational availability focused diagnosis of complex nuclear power systems. Expert System with Applications, **17** (1), (1999), 21-32.
13. Arroyo-Figueroa G.: Probalilistic Reasoning with Temporal Nodes and Its Application for Diagnosis and Prediction. Ph D. Thesis (In Spanish), Computer Department ITESM Campus Morelos, (1999).

A Complete Teamwork Model in a Dynamic Environment

M. Goyal and N. Parameswaran

Department of Information Engineering
School of Computer Science and Engineering
The University of New South Wales
Sydny 2052 Australia
{madhug, paramesh}@cse.unsw.edu.au

Abstract. A team is a task-oriented group, its behavior is constrained to eliminate actions that are not essential to task achievement. The differences of individual beliefs, intentions and plans cannot be accomplished, yet team demands that these different insights be combined or integrated such that agents act as one. The central hypothesis in this paper is that for effective teamwork, agents should be provided explicit time lines and an underlying model of teamwork based on a concept of team knowledge. The model also emphasizes on the team agent architecture that contains multiple level belief modules (team, group, society, and individual beliefs) as its core to allow robust and coherent team behavior. The application and implementation of this model to a virtual fire-fighting domain has revealed a promising prospect in developing team agents.

Keywords: Multiagent Systems, Multiagent Planning

1 Introduction

Multi-agent systems require coordination of resources with distinct expertise to perform complex tasks effectively. Recently the concept of teams is evolving into a dynamic multifaceted entity; computer scientists in distributed artificial intelligence study its complicated nature. According to some philosophers and sociologists, a team is group of people with a high degree of interdependence geared toward the achievement of a goal or completion of a task. One of the most popular and contemporaneous approaches of teams is that, the joint action by a team does not depend on simultaneous and individual actions, but it is based on team goals, mutual beliefs and joint commitment [3]. Other popular theories in this area are based on SharedPlan [5], joint responsibility [7] and intentions for multi-agent systems [13]. Teamwork is becoming increasingly popular in multiagent environments, such as, real-world battlefield simulator [14], industrial control [7], human machine interaction [9] and synthetic RoboCup soccer [8].

In this paper, we have developed an explicit model of teamwork. Our idea of team is based on knowledge attribute i.e. agents can not form team until they have definite knowledge about forming a team. A team agent is a system the behavior of which is directed towards implementing a specific state of the world i.e. is a goal-governed

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 718—728, 2000.

system. A team event occurs when two or more agents intentionally achieve a common goal, that is, a goal with regard to which the agents depend on one another. The key here is to recognize that when an agent participates in a team activity, and hence the agent must be provided with explicit model of teamwork. In other words, a team event is multi-agents plan i.e. a plan that necessarily requires more than one agent for it to be accomplished successfully. In its fullest sense, a team plan occurs when agents have definite knowledge that they depend on one another to achieve one and the same goal. Unfortunately, in implemented multiagent systems, team activities and the underlying model of teamwork is often not represented explicitly [7]. Instead, individual agents are often provided individual plans to achieve individual goals, with detailed precompiled plans for coordination and communication. Most of the research in multi-agent systems is focussed on how individual agents behave reactively and deliberately in complex and dynamic domains, such as, PRS [6], BB1 [12], RAP [4], Soar [10].

Section 2 provides the brief description of the fire world domain. Section 3 provides the conceptual framework in which some important concepts and ideas of teamwork are explained in the form of agent architecture. Section 4 describes the dynamic team formation. Section 5 provides the descriptions that how team plans are represented. Section 6 shows how these ideas are implemented and evaluated in a fire-fighting domain. In section 7 we discuss some experimental results. Section 8 and 9 reviews the pertinent literature and conclusion.

2 Fire World Domain

We have implemented our team ideas on a simulation of fire world *FFAGENTS* using a virtual research campus. The idea of simulated fire world was first given in Phoenix [2], which is a real time, adaptive planner that manages forest fires in simulated environment. The virtual campus is implemented using C++ on Windows95/NT platform, where more than 40 agents share the world via network. Part of this world that is related to our example is shown in figure 1. *FFAGENTS* is a dynamic, distributed, interactive, simulated fire environment where agents are working together to solve problems, for example, rescuing victims and extinguishing fire. The fire world *FFAGENTS* that we have considered in this paper consists of a large number of objects (of the order of hundreds) and several agents. Agents are autonomous and heterogeneous. Objects in the fire world include walls, buildings, furniture, open areas and LPG gas tanks. Our world is different from others' (like Air Combat [14] and RoboCup [8]) in respect that problems posed to the agents and the changes in the environment are not only caused by the actions of other agents but also by the changes the objects themselves undergo in the world (caused by the fire). In a world such as this, no agent can have full knowledge of the whole world. Humans and animals in the fire world are modeled as autonomous and heterogeneous agents. While the animals run away from fire instinctively, the fire fighters can tackle and extinguish fire and the victims escape from fire in an intelligent fashion. An agent responds to fire at different levels.

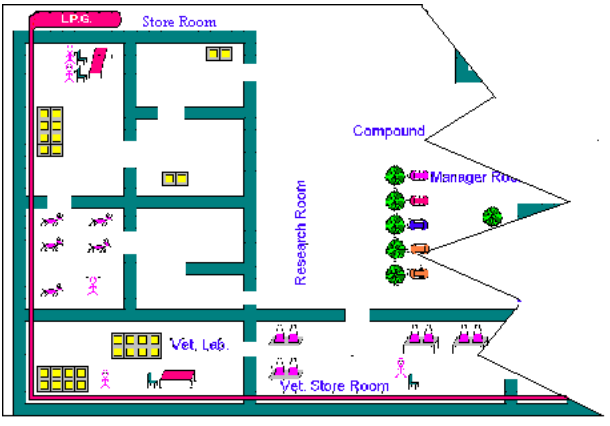


Fig. 1. A Sample Fire World

At the lower level, the agent burns like any object, such as chair. At the higher level, the agent reacts to fire by quickly performing actions, generating goals and achieving goals through plan execution. This world contains all the significant features of a dynamic environment and thus serves as a suitable domain for our team agents. Agents operating in the domain face a high level of uncertainty caused by the fire. Agents in the fire domain do not face the real time constraints as in other domains, where certain tasks have to be finished within the certain time. However, because of the hostile nature of the fire, there is strong motivation for an agent to complete a given goal as soon as possible. When a plan fails, the team agents must consider alternative plans.

3 Explicit Model of Teamwork

The problem of modeling the activity of team of agents is a combination of two sub problems: the first is the modeling of the team itself [15] and the second are the modeling of the team activity [6]. Instead of relying on *joint intentions* framework [9], we argue that team is an attribute of participating agents i.e. it is a combination of attributes like goals, plans, actions, intentions, knowledge, group, society etc. In our team model, each agent deliberately supports the team. Every agent in team is conscious of the team activity. Thus a team model in an application is an explicit association between agents and the world. Every agent in a team should have a meta attribute called Know (x). Know (x) has two components.

1. Model of x – This will be a name of x and details of x which are adequate enough for the description of the behavior below.
2. Behavior – It can be both executional behavior and conversational behavior.

Thus the maintenance of team based systems is concerned with the team knowledge about the relationship amongst different agents. For example, when a company of firefighters put out fire together, each team member is well aware of the team activity – each individual is not merely putting out fire on its own, while merely

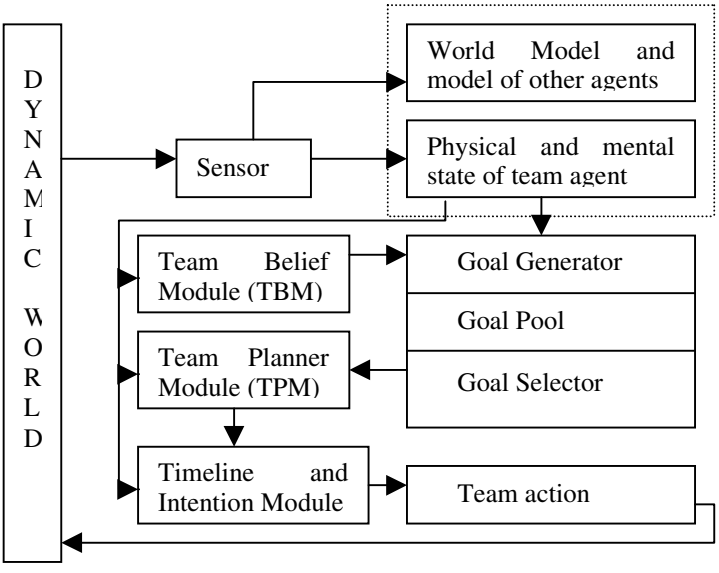


Fig. 2. Team Agent Architecture

coordinating with others. Also the behavior of agents in a team is a reasonably consistent pattern based on the dynamic integration of the sum total of an agent's knowledge, beliefs, goals, intentions and plans. The agent architecture shown in figure 2 is designed for intelligent agent executing cooperative team goals in a multi-agent environment. To work cooperatively with other agents, a team agent must have a self-model (Physical and Mental State of agent) and model of other agent. In our architecture, we propose four modules for a team agent, team belief module (TBM), team plan module (TPM), team goal module (TGM), and time line and intention module (TIM). The team belief module (TBM) consists of facts that the agent believes to be true about the world. Beliefs in general include beliefs about goals, plans, beliefs about other agents' beliefs and intentions. A maturing team moves toward becoming a social system in the sense that it is a group of people who interact with an accepted structure in order to achieve a goal. Thus a major function of team beliefs is to provide the necessary background knowledge in team plan execution, so that a flexible, efficient and coherent teamwork.

In team goal module (TGM) module, the goal generator uses different types of rules to generate the different goals from goal pool. The team plan module (TPM) specifies how a goal can be decomposed into team activities. The set of such plans for an agent is also called the *plan library.* A team can execute a plan only if all its members have the same plan. The purpose of a time line and intention module (TLIM) is to schedule the team activities. All the team activities are scheduled in a particular sequence or pattern. A team agent picks up the most recent goal/activity, expands it dynamically into low level details and inserts them into time line for execution. When agents form a team, problems emerge regarding the representation and execution of actions. Thus the time line module helps to formalize complex team plans into individual team actions.

4 Team Formation

Impetus for team formation may arise from the world and a particular domain or from agent themselves. Having identified the potential for team action with respect to one of its goals, a team agent will solicit assistance from some group of agents that it believes can achieve the goal. In any instance, it grows out of awareness that the demands of the work to be done require a use of a system, whereby a variety of specialised knowledge and skill can be brought to bear in a highly correlated fashion on the problem at hand. In a multiagent system, a particular domain recognizes appropriateness of the team model for the task at hand; set up the requirements in terms of agents, role designation, and structure. The major determinant in selection of team agents should be the goal for which the team is formed and the kind of knowledge and skill needed to achieve that goal.

Tasks do not remain static. Throughout the phases of teamwork, aspects of the issue or problem continue to change, as new facets become apparent. Some initial ideas may seem less applicable, or the implementation may be more complex than first predicted. Thus the state of team keeps on changing continuously in cyclic fashion from team formation, to team activity and to team dissolution during the lifecycle of a team. Thus, we propose a dynamic team formation model, in which we consider initially the mental state i.e. the beliefs of all the agents is same. In order to select a leader of the team, we suggest select that agent whose mental state is more advanced than the other team agents in a particular situation. We select an agent as a leader whom in a particular situation

- Has knowledge about the state of other agents.
- Can derive roles for other agents based on skills and capabilities.
- Can derive a complete team plan.
- Can maintain a team state.
- Has unified commitment towards team goal.

Our strategy for team formation requires commitment from the team members towards the joint goal and selection of team leader before the team is formed. We propose an algorithm to form team called *communicate-commit-and- select* that iterates through the number of steps until the team is formed.

- Suppose there is a fire, the agents start broadcasting message to each other “Let us form a team”.
- Agents form a team if all or more than three agents agree by saying, “Yes”.
- Then, they nominate some agent as their leader. The leader is selected according to the capabilities listed above.

In case the situation changes, the leader becomes redundant, and then they again go through the steps of team formation and select a new leader based upon the above conditions.

5 Representing Plans for Team Activity

Our team-planning module is combination of both *reactive and deliberative planning*. The dynamic nature of domain and issues of replanning forces the agents to plan reactively. Also the idea of team leader making decision requires a greater focus on deliberation than reactivity. The more focus on deliberation has led us to draw substantially from the classical planning literature. Our team planner uses hierarchical task-decomposition technique for the team task decomposition by decomposing abstract tasks into a set of sub team actions. A team plan is a way or approach that the team agents choose and agree on to do their task and achieve their goal. It is both an abstract structure /recipe and a set of complex mental attitudes [11][5]. Thus team tasks may be abstract or primitive. Abstract tasks may be decomposed into an ordered set of more specific tasks. We represent plans as Time Line Structures (TLS). Traditionally, time line structures are viewed as a collection of primitive actions, which are temporally ordered. Planning involves the construction of a time line, which gives detail of sequential list of actions, which guide from an initial state to final desired state. Suppose there are two agents (A and B) in a team. The time line structure for the team as well as of the agents is shown in figure 4. The team action T1 is further divided two actions a1 and b1. Similarly team action T2 is divided into actions a2 and b2. The team action T*n* consists of future actions a*n* and b*n*, which are shown dotted because of unpredictability of the environment.

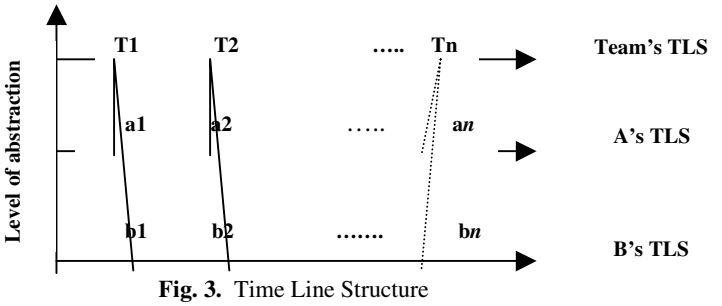


Fig. 3. Time Line Structure

However, for agents residing in a large and highly dynamic world, the time line structure enumerating all basic actions becomes too long and is difficult to reason with and maintain. Moreover, the agent often has only a partial knowledge of the world and its action execution capabilities, and it may not always be possible to fully develop time lines to their finest details. The plan structure consists of both team actions and sub team actions that are yet to be planned for. Agents operating in real-world applications must be able to notice and respond to unplanned situations. The range of possible events in the world is vast, while an agent is constrained by limited resources and limited computational speed. Because of these limitations, the agent is unable to build complete plans within arbitrary deadlines. Thus in most of real world applications, a team usually has only a partial plan at the initial stage because of dynamics and unpredictability of agent's environments. To deal with such reactive situations, we suggest incorporation of *IF-THEN rules* in the partial plan [1]. This problem solving strategy provides a framework that allows interleaving planning and execution. Thus replanning may be triggered in response to unexpected events. Thus

the planning in real world is a dynamic process. A team plan specifies a series of team activities that the agent needs to perform in order to achieve or maintain the team goal. The team activities within the team plan must eventually rest on actions of individual member's or subgroup; the collaborative team actions are usually highly abstract and consist of actions for individual members or subgroups (Figure 3). Thus a solution to a team problem can be specified in three ways:

- Plan-Execute Method: It involves deriving a plan, and executing the plan. This approach is used when the world is reasonably stable during planning and plans execution.
- Behavioral Method: It involves executing a set of rules. This approach is used when the world is changing too fast to do planning.
- Combination Method: It involves adapting a combination of the above two methods.

Typically, the team task module manages the actions of team agents. When a new fire is reported, *FFAGENT* retrieves an appropriate plan from the plan library and places it on the timeline. Information about the resources and sensory input are stored in the world model and is used to help the team agent to select the appropriate plan. At any time during this process, sensory data can trigger reactive actions, which will cause the team planner to modify its plan (such as moving away from the fire ahead) by incorporation of some rules. Thus we use combination method to solve team problems in a dynamic world. Using the behavioral method, the solution to the *TeamPutOut(fire)* according to the type of fire is shown in figure 5.

```
TeamSense(MediumFire) ->
[TeamClearObjects(MediumFire),TeamChoose(SubFire),
TeamPutOut(Subfire)]

TeamSense (SmallFire) ->  [TeamChoose(Ai),
AgentExec(Ai, SmallFire)]

TeamSense (NoFire)    ->  [TeamTerminate (AllActivity,
TeamGoOut]
```

Fig. 4. Solution for putting out fire (Behavioral method)

The right hand side of each rule is in the form of a plan. This plan must be loaded on the TLS and executed. But each action the plan is actually a complex goal. We need to specify solution for each goal the way we did for the goal *TeamPutOut (fire)*. The plan based solution for *TeamPutOut(subfire)* for a team of two agents (each has the capability of putting out fire in one square) will be like shown in figure 6.

```
1. TeamPlan=[TeamPutOut(sq1,sq2),TeamPutOut(sq3,sq4)…]
2. TeamExecute(TeamPlan)
```

Fig.5. Solution for putting out fire (Plan based method)

6 Implementation

We have developed implementation and experimentation in fire fighting in virtual research campus to verify our ideas about the team problem solving. Here we describe the approach taken to the modeling of teams and team behavior a part of the modeling of fire fighting missions. This work is part of the development of the Fire Fighting Agents called *FFAGENTS*. *FFAGENTS* is a dynamic, distributed, interactive, simulated fire environment where agents are working together to solve problems, for example, rescuing victims and extinguishing the fire. The FFAGENTS is *split into three main separate programs: FFplanner, DBserver and MsgServer.* Each *FFplanner* program is responsible for one fire fighting agent (or FFman). The *FFplanner* program can control the movement of fire fighting agent by sending a command message (which contains the required action to be performed) to *DBserver* for execution. If the action is performed successfully (by the *DBserver*), the *FFplanner* will receive an updating request message (from domain server) to update its own local world. The domain server is split into two parts: *Dbserver* and *MsgServer*. The *DBserver* is responsible for maintaining the world domain knowledge, performing several system critical tasks (such as the fire simulation process) and actions that are requested by *FFplanner.* When a requested action is performed successfully, *DBserver* will update the world domain knowledge base and send a domain-updating message to *MsgServer.* Upon receiving this message, the *MsgServer* will re-direct this message to all shared agents requesting them to update their own local world.

We have found team problem solving is fast, effortless and is propitious for coordinated activities between agents. Whereas the problem solving without a team is slow, laborious and can lead to conflicts between agents. In our example, fire occurs in the storeroom where four fire fighting agents (A, B, a, b) quickly form a team, adept roles, select a team plan to handle the fire emergency (figure 7). In order to evaluate the effectiveness of the team formation and team-planning scheme we have proposed we carried out several experiments under different fire situations. The goal of the agents in our experiment is to form team to put out the fire. In order to combat the fire, a fire-fighting agent must be supported by another fire fighting agent. In the virtual fire world *FFAGENTS*, the command [teamup] allows a fire-fighting agent (leader) who is holding a hose to be supported by another fire fighting agent (supporter). It is of crucial importance for a fire fighter to be supported as quickly as possible in an emergency situation. During execution, [teamup] automatically scans though the list of fire fighting agents and compares the distances of each eligible candidate with respect to the intended leader. The closest candidate, who must not be a leader or currently supporting another leader, will be requested to support the intended leader.

Two fire fighting agents ‘A’ & ‘B’, each holding a hose, required support from other fire fighting agents. There was one eligible supporter near the two agents. One of the leaders issued a request and a supporter responded. While supporter ‘a’ was moving towards ‘A’, agent ‘B’ issued a request for support. At that time, supporter ‘a’ was closer to ‘B’ than another eligible supporter ‘b’. However, as supporter ‘a’ has already agreed to support ‘A’, supporter ‘a’ did not acknowledge ‘B’s request. Instead, supporter ‘b’ responded and moves forward to support agent ‘B’.

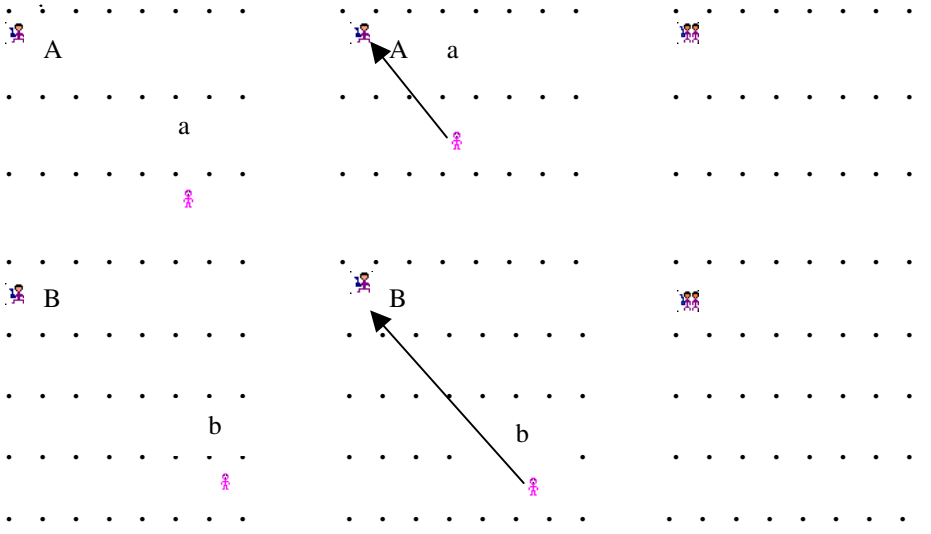


Fig. 6. Scenario for testing the [teamup] command

7 Experimental Results

Our experiments also concentrate on the evaluation of the significant level of coordination among team members. Agents based on our approach in our domain contain 40 plans and 1000 rules, with roughly 20% dedicated to our explicit model of teamwork. Although all teams represent a collection of people who must collaborate, to some degree, to achieve common goals, there is difference in the amount of collaboration required. There are some teams that are not required by the nature of the task to work closely together all the time. Such teams are called *low teams*. Unlike the low teams, there are others who must meet together more regularly and whose work requires more consistent coordination. Such teams are called *average teams*. In some groups constant teamwork is required by the nature of the task. Every person depends on everyone else. All of the tasks are highly connected and members cannot do their respective work without others doing theirs. Such teams are called *high teams*.

Fires of varying sizes were created in the world. A small fire refers to a single occurrence of fire in the world. A medium fire refers to two or three occurrences of fire in the world. A large fire refers to four or more occurrences of fire in the world. We ran our experiments using three different types of team: the low team, the average team and the high team. Figure shows the performance of the team based on the different degrees of coordination and types of fire.

From figure 8, we observe severer the situations (from small fire to large fire), the more efforts the team spends in achieving the same goal (put out the fire). In the meantime, the chances of plan failures increase with as the world becomes more hostile. This is understandable since large fire creates more difficulties, and poses more challenges for team's planning and plan execution. We note that when the fire is small, the team plan seems to do well, because there are less chances of plan failure. So the number of rules fired are also less.

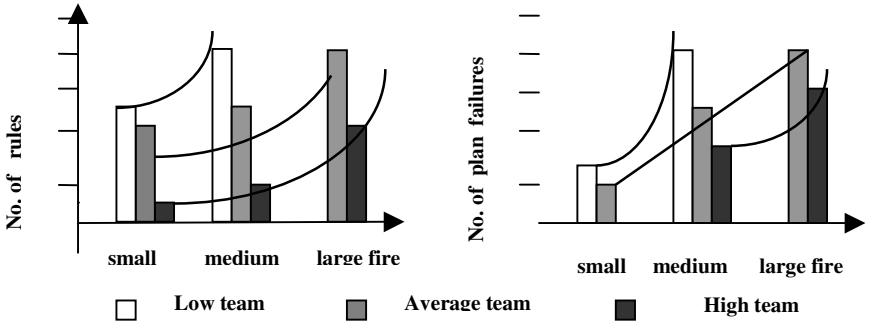


Fig. 7. Performance comparison

8 Related Work

Levesque gave a general model of teamwork, which is based on the joint intention concept [3]. The model specifies that the members of a team should jointly commit to the team goal. Wherever the team goal state changes, i.e. the team goal is achieved, unachieved or irrelevant, a member of the team who found this fact should be responsible for such private discovery known to all team members. In our approach, agents do not need to do this, unless the team/group require them to do. There are applications where agents are required not to communicate such discoveries. In such cases, the team/group beliefs will explicitly enumerate the list of actions an agent is required to do or not to do. Barbara J. Grosz's study of teamwork is a model of collaborative SharedPlan, in which the team members are required not only to be committed to the success of the team achieving the goal, but also follow the team plan when doing so [5]. That is, the team plan here brings further constraints /guidance to bear the team members' activities. However, as we explain in this paper, joint intention and SharedPlan alone are not sufficient for the successful execution of the team plan, but a team should have defined roles i.e. a team leader and team expert for successful execution of a team plan. In applications, Milind Tambe's teamwork model (the STEAM system) has been successfully applied in the virtual military training and RoboCup soccer game domains [14]. The STEAM system draws on the joint intention concept, and is built based on Soar architecture, and employs reactive planning. The agents in STEAM system seem to be more homogenous in terms of their capabilities. The hierarchy of operators /strategies is known to every agent in the system, including the opponents. There is little chance for agents to deviate from the standard, well-defined team strategy. Such models may very well apply to the domains like military and Robocup Soccer game, but it is not appropriate in many other domains like fire-fighting domain, where different types of highly autonomous agents with heterogeneous capabilities inhabit and interact with each other. Nick Jennings applied agent teamwork in the industrial electricity management system [7] using a framework called joint responsibility based on joint commitment to a joint goal, and joint commitment to a common recipe. These two types of joint commitments (a variation of joint intention) are claimed necessary because different actions are taken when the joint commitments are dropped. However joint responsibility model seems to be limited to a two level hierarchy of a joint team goal and a joint team plan, which is different from our approach.

9 Conclusion

Concluding our work in this paper, we explicitly present a complete teamwork model, which is used to coordinate agent activities and solve problems in dynamic environments. We also introduce the concept of multi-level agents' beliefs to enforce flexible, coherent and collaborative teamwork in a multi-agent dynamic world. It is believed that in a dynamic domain where agents are autonomous and heterogeneous, there is a need for leader agents providing plans to general agents. In addition, there is also the necessity of introducing individual plans at certain level of abstraction into the team plan. We argue that because of the dependency of the agents' actions and the risk of individual action failure, the team plan should also specify (wherever necessary) the way how an individual agent will execute the action within the team context, i.e. giving the individual plan within the team plan, so that all team members are explicitly aware of it and thus form the corresponding meshing plans. Although we have stated that the role allocation is dynamic, still there can be conflicts and negotiations amongst team agents. This could be interesting source of argumentation in the future.

References

1. Ambros-Ingerson, J.A., and Steel, S. Integrating Planning, Execution and Monitoring. In Proceedings of the National Conference on Artificial Intelligence (AAAI-88).
2. Cohen, P.R.; Greenberg, M.L.; Hart, D.M.; and Howe, A.E. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3), 1989.
3. Cohen, P.R., and Levesque, H.J.Teamwork. *Nous*, 35, 1991.
4. Firby. J. An investigation into reactive planning in complex domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI), 1987.*
5. Grosz, B.J., and Kraus, S. Collaborative Plans for Complex Group Actions. *Artificial Intelligence* 86 (1996), pp269-357, 1996.
6. Ingrand, F.F.; Georgeff, M.P.; and Rao, A.S. An Architecture for real-time reasoning and system control. *IEEE Expert 7(6), 1992.*
7. Jennings, N. Controlling Cooperative problem solving in Industrial Multi-agent Systems using joint intentions, *Artificial Intelligence* 75, 1995.
8. Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; Osawa,E. Robocup: The robot world cup initiative. *In Proceedings of the IJCAI-95 Workshop on Entertainment and AI/Alife, 1995.*
9. Lochbaum, K.E. Using Collaborative Plans to Model the Intentional Structure of Discourse. Ph.D. thesis. Harvard University, 1994.
10. Newell, A. *Unified Theories of Cognition*. Cambridge, Mass.: Harvard University Press,1990.
11. Rao, A.S. A unified View of Plans as recipes. Technical Report 77. The Australian Artificial Intelligence Institute, 1997.
12. Hayes-Roth, B., Brownston, L., and Gen, R.V. Multiagent collaboration in directed improvisation. In *Proceedings of the International Conference on Multi-agent Systems (ICMAS-95),* 1995.
13. Singh, M.P. Intentions for Multi-Agent Systems. MCC Technical Report No. KBNL-086-124, 1993.
14. Tambe, M . Agent architectures for flexible, practical teamwork. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*,1997.

Towards a Personalized Web-Based Educational System

J. G. Boticario and E. Gaudioso *

Dpto. de Inteligencia Artificial
Spanish National University for Distance Education (U.N.E.D.)
Senda del Rey, 9
Madrid, Spain
{jgb,elena}@dia.uned.es

Abstract. The Internet is rapidly becoming the main link to the university for distance learning students. Taking as given both the definition of a model of the education and that of the Internet specific didactic material, this article concentrates on the description of the objectives, the structure and the functionality of an interactive system capable of adapting to the information and communication needs of the different types of user. The application we have developed is the result of an effective combination of techniques used in intelligent tutoring systems, adaptive hypermedia programs and learning apprentice systems for software personalization. In the initial stages, this application is being used for the personalization of the presentation of the exercises' part of the machine-learning courses at the Computer Science School of the UNED (The Spanish National University for Distance Education).

Keywords: Distance learning, Adaptive Hypermedia, Web-based Adaptive Educational Systems, Machine Learning, Learning Apprentice Systems, Intelligent Computer-Aided Instruction.

1 Introduction

The "systematic use of communication media and technical support" [12], specially the services offered over the Internet, is becoming the main contact link with the university for both educational purposes (e.g., for the student to discuss his/her problems with other students or with the teacher irrespective of his/her physical isolation) and administrative purposes, lightening the load associated with compiling his/her academic record.

Considering the varied nature of the characteristics of the distance learning student and the dispersion of the sources of information to be used (news, mailing lists, the different kinds of Web pages such as those of the institution, those containing course information, practical exercises or assessment tests, the lecturers' personal pages, FAQs, etc.), interactive systems which adapt to the information

* PhD grant from UNED

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 729–740, 2000.

and communication needs of each student and are capable of continually improving on the quality of the answer to the user, both in terms of efficiency and in effectiveness, can be seen to constitute a suitable approach [3].

The pourpose of this article is to present the development of an interactive Web-based system that adapts to the different needs of each type of student using machine learning techniques. We describe the architecture of the system, its components and current functionality. Taking this description as a starting point, we then comment on the trials carried out up until now involving the personalization of the exercises' part of UNED's Computer Science School machine-learning courses and the Artificial Intelligence Department postgraduate courses. In these trials, the system serves as a support for focusing the student's attention on some of the most interesting elements for the user (exercises which may carried out, questions which must be answered, web pages related to the chosen course, etc.), in accordance with the student's academic record and the current state of his/her knowledge of the problem at hand.

2 Objectives

This application, based on a model whose principal objective is to focus the teaching on the student performance, rests on five basic pillars:

1. From the point of view of the psycho-pedagogical model of teaching-learning, its main objective is to encourage *significant and active learning*, in which the main protagonist is the student. In order to accomplish this, the *natural learning model* is used. This model consists of the following steps [14]:
   a) Raising interesting objectives.
   b) Generating questions which are useful to respond to the established goals.
   c) Processing answers to the questions raised.
2. It is a distance learning support system, which aims to ensure rapid, efficient and personalized access to the relevant information.
3. It is based on a multiagent architecture, which combines the approaches known in the machine learning literature as *learning apprentice systems* and *adaptive interfaces*. The *adaptive interfaces* interact with the user by constructing a model of the latter based solely on experience of actual use of the system by that user [13]. The system agents are thus able to modify and extend that model (knowledge base) in accordance with the following principle:

   One can construct interactive assistants that, even when they are initially uninformed, are both sufficiently useful to entice users and able to capture useful training examples as a by-product of their use [7].
4. It works through access to the educational services available on the Internet, it is transparent to the student and no additional specific software is required.
5. The system is based on a specific education management proposal [1] for distance education over the Internet. The didactic material supplied follows the guidelines that are considered appropriate for this kind of education [2].

3 A Distance Learning Interactive System

At the moment, the system combines techniques applied in intelligent tutoring systems (ITS) [15], adaptive hypermedia programs (AH) [4] and learning apprentice systems (LA) [7], and it falls into the category of the so-called *Web-based Adaptive Educational Systems* [5]. The system performs tasks having the following goals: to correct the user's behavior (ITS) and provide effective support for the different decision tasks on the Web (LA and AH). The intention of the learning apprentice system approach is to expand the initial knowledge base in order to reduce the effort required in user decision-making (adaptive hypermedia).

The system is designed to satisfy the information, cooperation and communication needs of a virtual community of users with shared educational interests.

The student-assistance task is implemented via an adaptive interface [13], which is based on a dynamic interaction focusing on the following elements:

- *Modifications in the interface*: The user perceives the dynamic aspect of the interaction through: different means of presenting information (highlighting, hiding, annotations), changes in the order in which the interface elements appear, changes in the level of interaction (opening new options according to the inferred user skill-level) and different descriptions of the information (see figure 1).
- *Adaptations in the domain:* exercise-solution assistant, personal study plan assistant, teaching-organization assistant for each course, problem-solving assistant for each subject area, communication-channel selection assistant, extra-curricular activities assistant (for both organizing and participating) and course-material assistant (for both management and modification). These tasks, in turn, are subdivided into curriculum sequencing, intelligent analysis of student solutions, interactive problem-solving support, example-based problem solving, etc. Each of these problems requires a specific solution.

Given the importance of those tasks requiring the manipulation of course-contents on the Web, we follow the model proposed by Schank in the ASK system [14]. In this model, contents are presented through a network of concepts (nodes) defined by the tutor and to be learned by the student. The edges represent transitions from one concept to another, to be executed according to the knowledge the student has acquired so far and according to his/her particular interests. These transitions are presented to the student in the form of questions to be answered or options to be chosen. The most significant types of node in this network are: development, introduction, objectives and exercise.

The main objective of the system is to carry out a suitable modeling of the user, from the a priori academic data available and his/her interaction history. In order to do this, machine learning tasks are carried out applying different generalization algorithms for each of the elements considered as being of interest.

Experience has shown that it is possible to carry out personalized interaction with the user in a transparent (without the need for specific software) and efficient manner, based on dynamically-constructed HTML pages and forms.

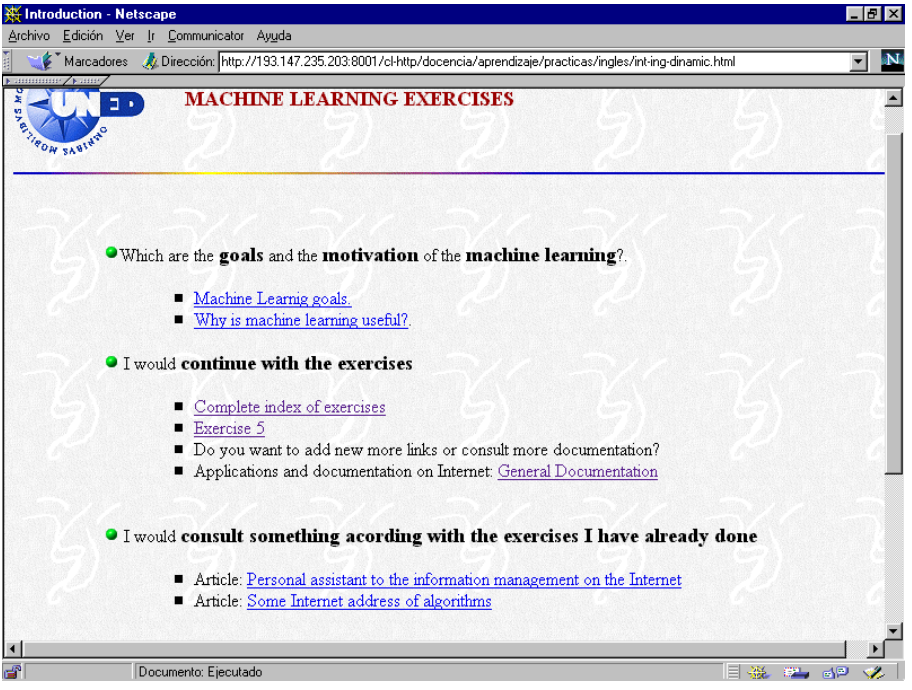


Fig. 1. Personalized exercices' home-page according to a previous user's interaction

As far as the format of the pages is concerned, the system also makes use of a XML version of the pages in order to separate the contents, the presentation, the behavior and the hypertext structure of the document. This technique allows the local user to interact with the document with no server connection (forms which check whether they have been completely filled in before being sent to the server, tests with on-line help given according to the user's response, etc.).

3.1 The Architecture

Considering the advantages of collaborative learning, the number of independent tasks involved, the dispersion of the existing resources and the unpredictability of the result (produced by the very nature of the system which endeavors to satisfy different user needs evolving over time), we have opted for a multiagent architecture; more specifically, a multiagent decision system organized as follows.

The two main components are interaction with the user and the adaptive module. The first is implemented by the interface agent and is in charge of the organized presentation of the different types of information according to design rules, which aim for the utmost usability. This module provides a single, integrated reply to the user. It is this module, therefore, which actually carries out the collaboration with him/her.

The adaptive module is made up of the following agents: user model agent, user modeling agent, material agent, pedagogical agent, contact agent, service identification agent, service agent, service modeling agent, and the coordinator agent. The first four provide the basic functionality of an ITS. The next four are useful for identifying, by *collaborative filtering*, the system services of interest to a user group. Finally, the coordinator agent broadcasts a problem instance description (user's request) to all the agents involved in that problem. This agent is also in charge of constructing a single response for the interface agent (see figure 2).

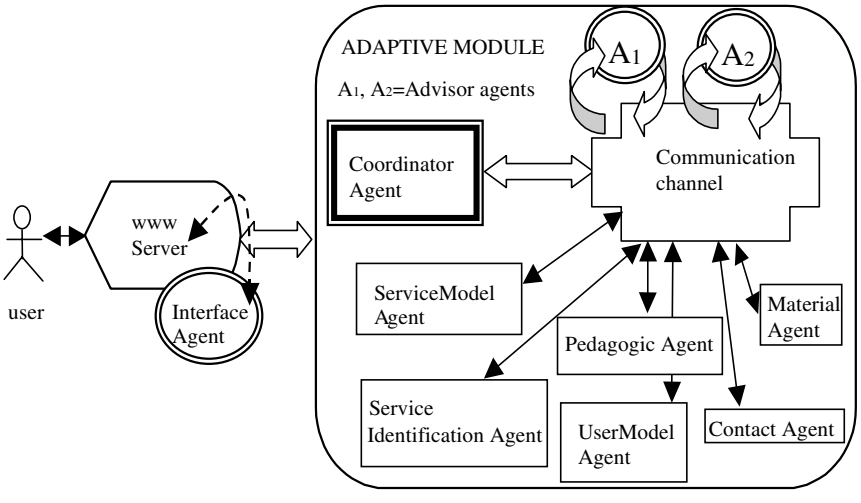


Fig. 2. System's general architecture

In more detail, the adaptive module consists of a dynamic multiagent system (agents join or leave the system) with weak coupling (they may or may not participate in each task). We have chosen heterogeneous agents in order to combine the solutions learned with different bias and corresponding to different generalization methods: C5.0, Naive Bayes, Progol, Backpropagation, Tilde and Autoclass for clustering. To implement this combination a special type of agent has been introduced: the advisor agent. Several advisor agents learn the competence range of each of the other agents. A task is only distributed to those agents that have been proved to be competent in that task. We thereby aim to establish a gradual process of agent specialization with concrete architectures for specific problems (see figure 3).

We distinguish two phases: application of learned knowledge and adaptation to the individual user through learning tasks. In the first, the interface agent makes a request to the coordinator. Depending on the request, this agent asks the advisors for the agents competent in the tasks involved in that request (the

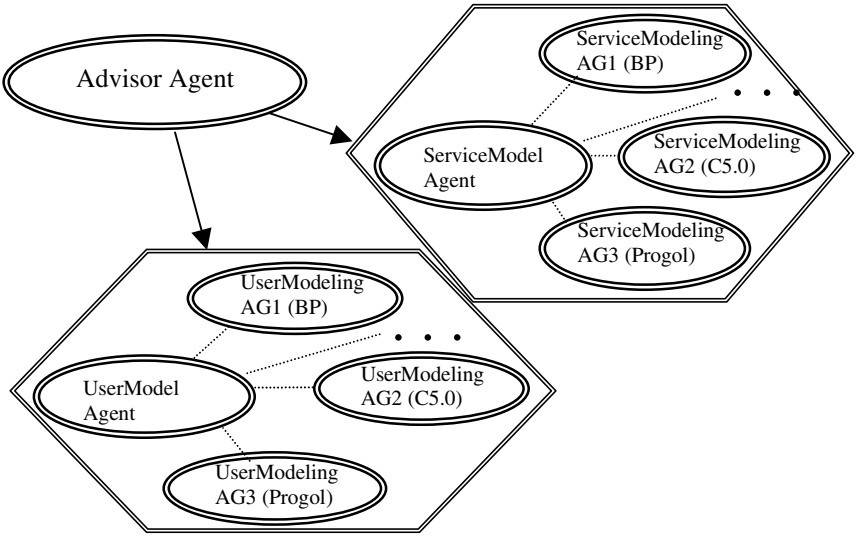


Fig. 3. Outline of the participants in the competence range learning task

intention here is to improve on other approaches that distribute the same task to every agent in the system [11]). In the second phase, when the interaction with the user has finished, with the available training examples (those that have been collected), the advisor agent learns the competence of each agent involved in the adaptation task: the model agent and the service agent. The learning task of the user model and that of the service model are implemented using the different generalization paradigms, which constitute the various modeling options: service modeling agent and user modeling agent. The service identification agent selects the services, which interest a significant number of users through *clustering* techniques.

Communication between agents is carried out using the KQML language [9] and the content of the messages is represented in KIF [10].

Taking into account our previous experience in the construction of learning apprentice systems [7], and given their effectiveness, we developed our system on THEO: an *integrated architecture* providing a generic programming environment in which different knowledge and inference representation mechanisms can be combined.

Initially, the system has a representation of all the domain entities together with their relations. For this, the corresponding ontologies, which include the entities, relations and functions used to model each of the system's components and each of the tasks that it carries out, have been defined. Among the most important of these are the dynamic construction of personalized pages, the updating of the student model and the services involved, the tracing of user interaction and the learning of the range of effectiveness of the advice given.

The concepts exchanged in the messages between the system's agents depend on the ontologies used, which in this case are specific to educational systems [6]. The system's knowledge base is made up of entities which define the application domain, the structure, the functionality and the definition of each of the agents, as well as the communication between them (see figure 4).

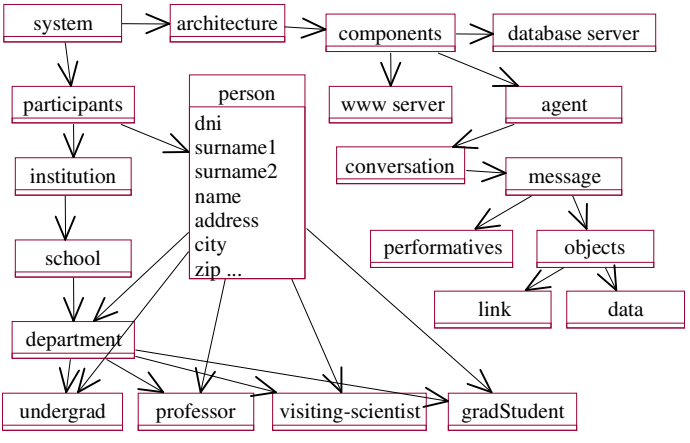


Fig. 4. Portion of general concepts' ontology

Each agent has its own knowledge base on THEO: the agents which perform learning tasks are able to extend their knowledge base dynamically, adding new entities and modifying the already existing ones, as the user interacts with the system (they adjust to what have been denominated as *learning apprentice systems*).

3.2 Interaction

As has already been mentioned, the user is aware of the dynamic aspect of the interaction through different presentations of the same information, changes in the order in which the interface elements appear, changes in the level of interaction and different descriptions of information. These changes depend on the task being executed at that moment.

The user interface uses different formats for different elements. When the advice cannot be integrated into the information shown (e.g., the upper part displays an access to news while the lower part shows other news or related mailing lists) or when the advisory information is contextual and does not directly affect the information shown (e.g., the upper part shows the course syllabus, and the lower part shows courses with related syllabuses). Otherwise, the system constructs the page together with the system's recommendations, adding

annotations, highlighting or hiding parts of the document, changing the order of the document's elements, etc.

In order to enhance the a posteriori learning of the user model, the system stores two information traces corresponding to the two aforementioned work areas. When the interface agent sends a request to the coordinator agent, it is the latter that divides and stores the aforementioned traces (e.g., if the student follows the link recommended by the system, it is considered a positive interaction, given that the system has advised the student well).

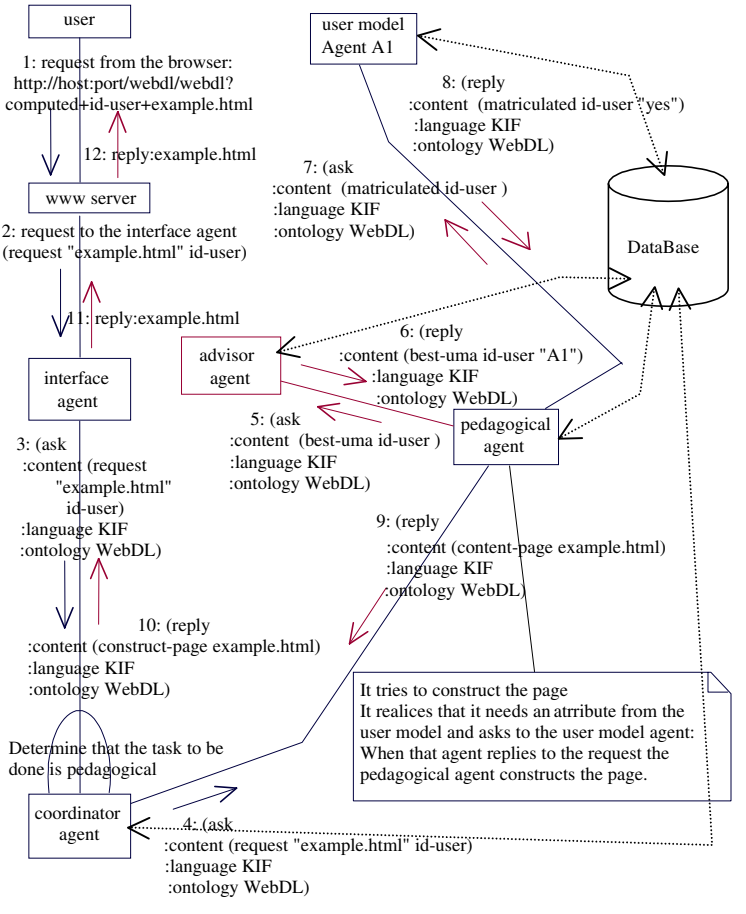


Fig. 5. Example of collaboration between agents in *adaptive navigation support*

In order to clarify some of the basic processes triggered when the user interacts with the system, detailed below is the communication sequence initiated when the student requests a specific resource via a link with a HTML page. Con-

sidering the diversity of tasks tackled (mentioned at the beginning of this section) and their complexity, we have decided to describe the *adaptive navigation support* task performed by the pedagogical agent in the subject exercise-material assistance. The rest of the tasks implemented already: curriculum sequencing (of all the course material), adaptive collaboration support (of the exercises' part of the course material) and adaptive navigation support (on all the pages constructed by the system) trigger similar interaction processes.

Each of the tasks performed by the system is modeled as if it were another entity of the system and it is defined in the knowledge base of the agent in charge of executing it, so this very agent could modify the definition of the task if it were necessary to best suit the student's requirements. The task that we are going to exemplify is adaptive navigation support. The idea of adaptive navigation support techniques is to help users to find their paths in hyperspace by adapting the way of presenting links to goals, knowledge, and other characteristics of an individual user [4].

When the student enters the exercises' part of the subject, a page appears with the description of a concept, exercise or example. The page is dynamically constructed by concatenating the information that we want to present together with a series of links that the student can follow after this page. One of these links will be the one that advises the system and it is therefore annotated in a special way; the student, in any case, can follow any of the suggested links.

Each of these pages is defined using an entity composed of a series of attributes defining them (page title, contents, author ...). The attributes used to construct a personalized student page in the adaptive navigation support task are `source-int`, `links-opc` and `next-link-advised`. The attribute `source-int` defines the path to the page text file desired, the attribute `links-opc` is a list of the links appearing to the student and the attribute `next-link-advised` contains the link for advising the student. The rest of the tasks performed by the system are similarly defined: the entities managing the task and the actions that the agent in charge of executing the task has to perform.

For each student request, the agent in charge of resolving it (in the case of the adaptive navigation support task applied to the exercise material it is the pedagogical agent) creates a page instance requested in the knowledge base. This instance stores some data on the student's reaction with the page, the student's degree of comprehension of the contents presented on this page, the student's answers if the page is a questionnaire, etc. These attributes make it possible to validate if the adaptation that appears to the student has been effective.

As we have said the attribute `next-link-advised` of the entity page example being accessed is the one that determines the link that the system is going to advise the student. This attribute is calculated according to a series of rules which can depend in turn on other entities in the knowledge base; for example they can depend on some user model attributes (`course-interest-level`, `matriculated` ...); these rules have a prolog rule syntax and can be included in the definition of an attribute in an entity on THEO [7]:

```
(exercise-4 *novalue*
 (GENERALIZATIONS (page-html))
 (ID-USR *novalue*)
 (LINKS-OPC ((‘‘OBJ’’ ‘‘Objectives’’ ‘‘obj.html’’ ... )
        (‘‘ACTV’’ ‘‘Activities’’ ‘‘actv.html’’ ... )))
 (SOURCE-INT ‘‘../www/webdl/exercise-4.html’’)
 (NEXT-LINK-ADVISED *novalue*
     (prolog.clauses (
         ((next-link-advised ?r ?value):-
             (id-usr ?r ?user)
             (matriculated ?user "no")
             (visited ?user ‘‘obj.html’’ ‘‘no’’)
             (eval ‘‘OBJ’’ ?value))
         ...
         ((next-link-advised ?r ?value):-
             (id-usr ?r ?user)
             (matriculated ?user "yes")
             (visited ?user ‘‘actv.html’’ ‘‘no’’)
             (eval "ACTV" ?value))))))
```

Each request that the student makes to the Web server generates a stream of messages between the system agents. When the student requests a specific page in the exercise mode, the Web server sends the request again to the interface agent which in turn passes the request on to the coordinator agent to decide which agent must be in charge of the request. Given the nature of the task that we are exemplifying, the agent, which will be in charge of constructing the page requested by the student, is the pedagogical agent.

Figure 5 shows the transfer of messages taking place when the student requests a specific exercise page. In this example, the rule associated with the attribute `next-link-advised` requires a user model attribute (`matriculated` which indicates if the student is matriculated or not in the subject). In this instance, the pedagogical agent does not have this datum and the request is therefore sent to the other agents. The user model advisor agent returns the identification of the competent user model agent, which in turn returns to the pedagogical agent the attribute value requested in accordance with its user model; this has been generated by the modeling agent with better results in the learning tasks performed with the different algorithms (C5.0, Tilde, Autoclass ...). When the pedagogical agent receives the attribute value it has requested, it can calculate the attribute value `next-link-advised`, construct the page and thus reply to the user's request.

3.3 The State of Development of the Project

To date, we have built the basic architecture of the system and we have used it to adapt access to UNED's Computer Science degree computer-learning course material.

The system tasks which we plan to implement are the following: curriculum sequencing, intelligent analysis of student solutions, interactive problem solving

support, example-based problem solving, adaptive presentation, adaptive collaboration support and adaptive navigation support. Of these, those that have been implemented are the following: curriculum sequencing (of all of the course material), adaptive collaboration support (of the exercises part of the course material) and adaptive navigation support (in all the pages constructed by the system).

At present, we have only experimented with potential users; these experiments have validated the proposed architecture in adaptation tasks previously described. Given that the academic year in our University begins in October, we expect to offer a comparison of the responses of the system in the different tasks, throughout this term.

4 Conclusions

The principal objective in this work is to centre the teaching on the performance of the student. In this article, we have presented the objectives, the structure and the functionality of an interactive system based on a multiagent architecture, to be adapted to the information and communication needs of the different types of user in the context of a specific organization of educational services available on the Internet suitable for the needs of UNED. We have demonstrated that personalized interaction with users/students can be accomplished in a transparent and efficient manner through the Web, without the need for specific software, our approach being based on dynamically-constructed HTML.

In order to prove the usefulness of the system and verify its possibilities, as the first step in our experimentation, we have developed a preliminary version of the architecture, which satisfies part of the required functionality and has shown its applicability in personalized access to the exercises' part of UNED's Computer Science School courses. For the moment, while awaiting more conclusive data from system use during the current academic year, we have discussed the sources of information which we have used for the learning tasks of the first working prototype and the framework in which they operate. In the near future, the application of the system will be extended to the rest of the Artificial Intelligence Department courses and services and to the rest of the Computer Science School services at such a time as these become available.

The personalization capacity of the system depends directly on the effectiveness of the design of the learning tasks. In our design we make use of: different combinations of classifiers in the generalization tasks [8], filtering of information on the elements used, automatic extension of the system knowledge base, individual and collaborative learning for the user models and learning of the services which are critical for the user community.

Acknowledgements. The authors would like to acknowledge the helpful comments of Simon Pickin and Anita Haney, arising in the course of his language revision of this article. We also thank the entire Artificial Intelligence Department for providing support for this project. We would like to expess our deep

gratitude to professor Tom Mitchell at Carnegie Mellon University for providing THEO for research purposes.

References

1. Jesus G. Boticario. Internet y la universidad a distancia. *A Distancia*, pages 64–69, Autumn 1997.
2. Jesus G. Boticario. Material didactico y servicios para la educacion a distancia en Internet. *A Distancia*, pages 70–76, Autumn 1997.
3. Jesus G. Boticario and Elena Gaudioso. Towards personalized distance learning on the web. In J. Mira and J.V. Sanchez-Andres, editors, *Foundations and Tools for Neural Modeling*, number 1607 in Lecture Notes in Computer Science, pages 740–749. Springer Verlag, 1999.
4. Peter Brusilovsky. Methods and techniques of adaptive hypermedia. In *User Modeling and User-Adapted Interaction*, pages 87–129. Kluwer academic publishers, 1996.
5. Peter Brusilovsky. Adaptative educational systems on the world-wide-web: A review of available technologies. In *Proceedings of Workshop WWW-Based Tutoring at Fourth International Conference on ITS (ITS'98)*, San Antonio, TX, August 1998. Mit Press.
6. Weiqin Chen and Riichiro Mizoguchi. Communication ontology for learner model agent in multi-agent architecture. In *Proceedings of the International Conference on Artificial Intelligence in Education (AI-ED99)*, Le Mans, France, 1999.
7. L. Dent, J. G. Boticario, J. McDermott, T. M. Mitchell, and D. T. Zabowski. A personal learning apprentice. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 96–103, San Jose, CA, 1992. Mit Press.
8. Tom G. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18(4):97–136, 1997.
9. T. Finin, R. Fritzson, D. McKay, and R. McEntire. Kqml as an agent communication language. In ACM Press, editor, *Proceedings of the Third International Conference on Information and Knowledge Management*, pages 64–69, October 1994.
10. M. Genesereth and R. Fikes. Knowledge interchange format, version 3.0 reference manual. Technical Report KSL-92-86, Knowledge Systems Laboratory, 1992.
11. J. Ignacio Giraldez, Charles Elkan, and Daniel Borrajo. A distributed solution to the pte problem. In Giuseppina C. Gini and Alan R. Katritzky, editors, *Predictive Toxicology of Chemicals: Experiences and Impact of AI tools, Papers from the 1999 AAAI Spring Symposium, TR SS-99-01*, pages 82–85. AAAI Press, March 1999.
12. J. Desmond Keegan. From new delhi to vancouver: trends in distance education. In *Learning at a Distance. A world perspective*, Athabasca University, Edmonton, 1982. International Council for Correspondence Education.
13. Pat Langley. User modeling in adaptive interfaces. In *Proceedings of the Seventh International Conference on User Modeling*, Banff, Canada, June 1999.
14. Roger C. Schank and Chip Cleary. *Engines for education.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1995.
15. Gerhard Weber and Marcus Specht. User modeling and adaptive navigation support in www-based tutoring systems. In *Proceedings of the Sixth International Conference on User Modeling*, pages 289–300, Chia Laguna, Sardinia, Italy, June 1997.

CASSIEL: Modeling Intelligent Agents for a Lifelong Learning Environment

Gerardo Ayala and Arlette Hernández

CENTIA*, Universidad de las Américas-
Puebla. Sta. Catarina Mártir s/n, 72820 Cholula, Pue., México.

Abstract. In this paper we propose the software components and functionality needed for the implementation of an environment that supports lifelong learning. The paper presents the requirements and design issues of intelligent agents that support lifelong learning in a community. A user agent assists the learner in the construction of her/his learning plan, and the configuration of discussion groups. An information agent is modeled in order to assist the learner in searching the location of information and knowledge considered relevant. A facilitator agent is discussed, designed for supporting the social construction of knowledge in the community. The paper covers the proposal for learner modeling and the construction of learning plans, needed for this kind of learning environments.

1 Introduction

In our global knowledge society learning is considered the source of sustainable and competitive advantage. We are responsible of our own learning and change. In this context learning is a *lifelong* activity, distributed and collaborative, based on the active interaction among members of a community, sharing and constructing knowledge.

1.1 Lifelong Learning Environments

In the area of artificial intelligence applied to education, research has been focused on the modeling of intelligent agents and CSCL (Computer Supported Collaborative Learning) environments [1]. Currently there is an increasing interest in the development of information technologies that support *lifelong learning*. Lifelong learning, according to the definition by the European Lifelong Learning Initiative (ELLI) [2] is:

"a continuously supportive process which stimulates and empowers individuals to acquire all the knowledge, values, skills and understanding they will require through their lifetimes, and apply them with confidence, creativity and enjoyment in all roles, circumstances and environments."

Dunlap [3] defines lifelong learning as:

* Research Center for Information and Automation Technologies.

O. Cairo, L.E. Sucar, and F.J. Cantu (Eds.): MICAI 2000, LNAI 1793, pp. 741–747, 2000.

"any purposeful learning that people engage throughout their lives...an activity...to gain greater individual self-fulfillment and to improve the quality of life for the individual and the emerging society."

Dunlap has proposed a Web-based Support System (WPSS) based on generative and intentional learning methodologies, which promote the development of metacognitive and self-directed learning skills, considered necessary for lifelong learning activities [3]. Generative learning considers that the learner has to take responsibility of her/his own learning, being an active agent creating knowledge in an organized manner in her/his community. Intentional learning implies a learner who determines her/his learning goals, being aware of her/his own learning and progress. With the E-study approach [4] Vries and Castelein propose the Internet as the technical infrastructure for lifelong learning, and situated learning [5] as the social infrastructure. Flexibility (freedom), integration (consistency and coherence), instructional interaction and information enriched (relevant curriculum information) are the principles that guide the instrumentation of the E-study concept. It is convenient to note the different interpretation of the term lifelong learning in the field of machine learning, where is considered a promising direction for a new generation of algorithms [6].

1.2 CASSIEL

At the Universidad de las Américas-Puebla we have been developing CASSIEL (Computer Assisted Intelligent Environment for Learning) as a prototype for a lifelong learning environment [7]. CASSIEL contributes with a new approach of agent-based learning environments designed to promote and support lifelong learning. The research issues in the CASSIEL project are:

1. determine the software components and functionality needed for a lifelong learning environment, and implement them for a learning community in the area of information technologies, formed by university teachers, researchers, students and software industry professionals.
2. determine the information needed in a learner model for lifelong learning, used in order to assist the learner in the construction of her/his learning plan (supporting intentional learning), and her/his participation in discussion groups considered of her/his interest.
3. assist the learner in the location of information and knowledge considered relevant for her/him in the web, according to her/his interests and learning attitudes.
4. support the social construction of knowledge in a virtual community, assisting the learner in the organization and presentation of her/his beliefs and knowledge to the community (supporting generative learning).

2 Intelligent Agents in CASSIEL

We consider an intelligent agent as a software component situated in an environment, capable of autonomous action, reactivity, pro-activeness and social ability

[8]. We propose three intelligent agents for lifelong learning environments, which have been modeled and implemented in CASSIEL:

1. *User agent.* Constructs and maintains a learner model as a set of beliefs about its user [8]. It cooperates with other user agents sharing the information of their learner models in order to make an autonomous configuration of discussion groups, and construct a proposal for the learning plan of each learner, considering her/his interests as well as the demands and capabilities of the community members. It promotes collaboration and provides adaptability.
2. *Information agent.* Autonomously navigates through the community web sites in order to provide information about new and updated material believed to be of the interest of the learner. It intelligently searches for information based on keywords, allowing the easy location of relevant information and knowledge.
3. *Facilitator agent.* Assists the learner in the organization of the knowledge and information provided by her/him. It assists the learner in providing her/his beliefs (comments and recommendations) to the community, participating in this way in the social construction of knowledge.

3 Supporting Collaboration and Adaptability

The satisfaction of the learning needs and demands of each individual is the essence of lifelong learning [2]. We consider adaptability to the learner as an important issue. The user agent in CASSIEL facilitates the adaptability and the collaboration of learners by constructing and maintaining a learner model.

3.1 Learner Modeling for a Lifelong Learning Environment

The model of the learner is represented as a set of beliefs hold by her/his user agent [9]. The process of learner modeling is implemented based on AI techniques of belief revision [10]. The learner model includes the following beliefs, generated or removed by a set of production rules:

1. Desires. Considering her/his personal requirements and proposals for participation in a common task or project.
2. Intentions. Considering her/his commitments to participate in projects or cooperate with members of the community.
3. Capabilities. Considering her/his fields of expertise and qualifications.
4. Interests. Visited web nodes.
5. Learner's ego development phase. Considering her/his navigation strategies.

It is known that a lifelong learner presents three phases of ego development [11] which represent different learning and collaborative attitudes. These phases represent the degree of maturity of the lifelong learner: 1) *conformist*: Looking for social acceptance, making relation to progress of peers and association of new material to existing concepts; 2) *conscientious*: Looking for self evaluated

standards, interpersonal interaction, learning goals consistent with self expectations and making reflection and introspection for future goal identification; and 3) *autonomous*: Presenting autonomy and independence, tolerance to ambiguity and self directed learning.

3.2 Group Configuration

Collaboration in a lifelong learning environment is essential due that learners construct knowledge by externalizing, sharing and integrating new ideas (justified beliefs) into prior knowledge [12]. The user agent makes use of the learner model for the automatic configuration of discussion groups. In order to configure automatically discussion groups in the community, taking into account the desires, interests and capabilities of their learners, the user agents cooperate by exchanging information of the learner models of their respective users.

3.3 The Learning Plan

A necessary task for the lifelong learner is to construct her/his personal learning plan [2]. Research results indicate that adults spend about 500 hours a year at major learning efforts. In 73% of the reported cases learning is planned by the learner her/himself [13]. In CASSIEL a learning plan is represented by a network of web nodes to be visited and commented by the learner. The arcs are links that represent the cognitive relations of part-of, generalization, analogy and refinement, organizing knowledge based on the implementation of genetic epistemology ideas in learning environments [14]. Each node contains information about:

1. demands of industry, from current desires of professionals.
2. research topics, from interests and capabilities of participating researchers.
3. students interests.
4. related knowledge, from the part-of, generalization, analogy and refinement links.

This information is provided by the community learners through their facilitator agents. The initial proposal of the user agent is a stereotype learning plan, constructed based on the learning plans from the community members, with the cooperation of their user agents.

4 Locating Relevant Knowledge

The information agent in CASSIEL is designed to guide the participants in the location of knowledge and information. The information agent navigates through the community web nodes looking for information based on a request from its learner. A request to the information agent consists of a selection or input of a keyword. The initial sites for the search request of a learner are obtained from the interests in the learner models of the user agents.

The strategy of navigation of the information agent through the structure of hyper linked web sites is based on AI techniques of heuristic search, being a variation of best-first search [10]. The search results are URLs and titles of web sites that appear sorted by *relevance* (quantity of related content based on the topic of interest in the html file) by *popularity* (frequency of visit by the community members) or by *newness* (promoting new web sites or web sites that have been updated since her/his last visit), according to the ego development phase of the learner. For those believed conformist, by popularity; by relevance for those believed conscientious and by newness to those believed autonomous. The information agent keeps its learner aware of new web nodes, as well as recent modifications in visited nodes. The agent autonomously starts a travel through the corresponding servers, checking the information at the heading of those web sites indicated as interests in the user's learner model, and notifies any updating after the last visit to that site via CASSIEL by its learner. The information agent also notifies the learner about the new comments and links provided by the community members via their facilitator agents concerning a web node (an educational material, a research project or a development at industry).

5 Supporting the Social Construction of Knowledge

Lifelong learning is an active process where the construction of knowledge is essential. The theoretical foundation of knowledge construction in CASSIEL comes from the work of Nonaka and Takeuchi where knowledge in an organization is constructed through the phases of externalization, socialization, combination and internalization [15]. A detailed description of the role of CASSIEL agents in the social construction of knowledge can be found in previous work [16]. One approach for knowledge construction in constructive web-based learning environments implies the development of new documents linked to previous ones, by the learners [17]. The facilitator agent assists the learner in her/his active participation in the community by:

1. providing the URLs of considered relevant web nodes, relating them by cognitive relations of part-of, generalization, analogy and refinement (externalization)
2. including her/his beliefs (comments and recommendations) about a web node (appearing as related comments to the site), or indicating the justification of beliefs based on additional information (socialization)
3. validating those justified beliefs about web nodes presented by other members of the community, providing comments and relations to other nodes (combination)

Prepare people to be lifelong learners requires an environment for learning to learn. Learning to learn is enhanced by focusing on the "big ideas" or "deep principles" involving concepts and theories that help organize knowledge [18] more than on detailed skills or technical information. This is the reason why comments into web nodes are considered important, since those beliefs, once

justified, discussed and validated according to prior knowledge, may become new concepts for the community.

The interaction with the user and the cooperation among the user agent, the information agent and the facilitator agent are presented in figure 1.

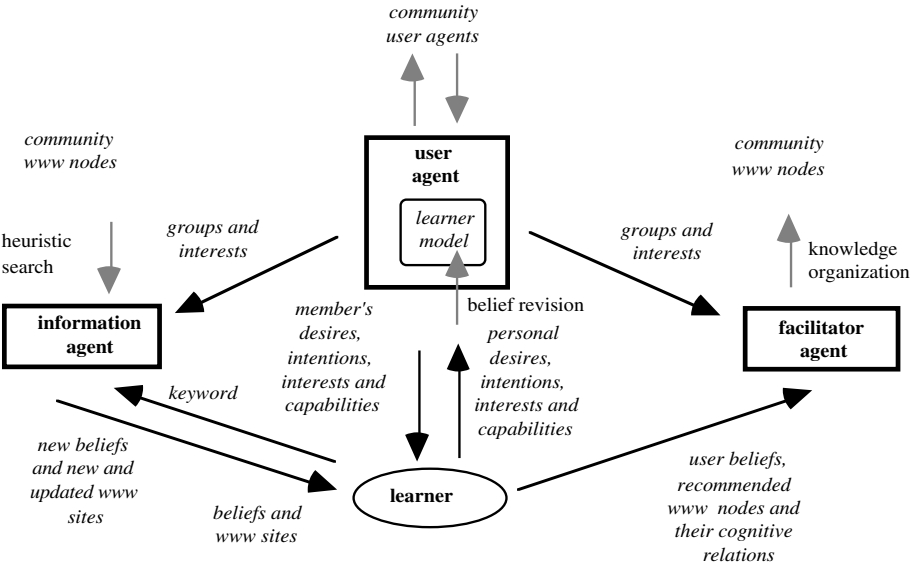


Fig. 1. Interaction among agents in CASSIEL.

6 Conclusions

In our society we learners need to maintain a learning plan, being aware of the demands and opportunities in our community. In this context, we have presented a framework for web-based environments that support lifelong learning based on agent and AI technologies, according to the requirements that these environments present. We believe that CASSIEL represents one more step towards the development of lifelong learning environments that promote the competitive advantage for individuals and organizations.

References

1. Ayala, G: Current Trends of Artificial Intelligence in Education. G. Ayala (Ed.), Workshop Proceedings, 4th World Congress on Expert Systems, ITESM, Mexico City, Mexico, (1998).
2. Longworth, N. and Davies, W. K: Lifelong Learning. Kogan Page, London (1996).

3. Dunlap, J. C: Revisiting the Web-based Performance Support Systems for Lifelong Learning: Learner-Centered Resource Development Tool. Proceedings of ED-MEDIA99, World Conference on Educational Multimedia, Hypermedia and Telecommunications, Seattle, USA, Betty Collis and Ron Oliver (Eds.) AACE (1999) 1140-1145.
4. de Vries, S. and Castelein, J: On our way to a Knowledge Community. Proceedings of ED-MEDIA99, World Conference on Educational Multimedia, Hypermedia and Telecommunications, Seattle, USA, Betty Collis and Ron Oliver (Eds.) AACE (1999) 397-402.
5. Brown, J. S., Collins, A. and Duguid, P: Situated Cognition and the Culture of Learning. Educ. Researcher 18 (1) (1989) 32 - 42.
6. Thrun, S: Explanation-Based Neural Network Learning: A Lifelong Learning Approach. Kluwer Academic Publishers (1996).
7. Hernández, A: Modelado de Agentes para un Ambiente de Aprendizaje en Internet. M. S. Graduate Thesis, Spring 1999, Dept. Computer Systems Engineering, Universidad de las Américas-Puebla, Mexico (1999).
8. Wooldridge, M: Intelligent Agents. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, Gerhard Weiss (Ed.) MIT Press (1999) 27-77.
9. Ayala, G. and Yano, Y: Learner Models for Supporting Awareness and Collaboration in a CSCL Environment. Intelligent Tutoring Systems. Proceedings of the conference ITS96, Montreal, Canada, Lecture Notes in Computer Science 1086, Claude Frasson, Gilles Gauthier and Alan Lesgold (Eds.), Springer Verlag (1996) 158-167.
10. Russell, A. J. and Norvig, P: Artificial Intelligence: a modern approach. Prentice Hall, New Jersey (1995).
11. Fales, A. W: Lifespan Learning Development. Lifelong Education for Adults, Colin J. Titmus (Ed.) Pergamon Press. (1989) 183-187.
12. Bos, E. S., Kikstra, A. and Morgan, C. M: Multiple Levels of Use of the Web as a Learning Tool. Proceedings of the ED-TELECOM 96, World Conference on Educational Telecommunications, Boston, USA, Patricia Carlson and Filia Makedo (Eds.). AACE, (1996) 31 - 36.
13. Tough, A. M: Self-directed Learning: Concepts and Practice. Lifelong Education for Adults, Colin J. Titmus (Ed.) Pergamon Press (1989) 256-260.
14. Goldstein, I. P: The genetic graph: a representation for the evolution of procedural knowledge. In Intelligent Tutoring Systems, D. Sleeman and J. S. Brown (Eds.) Academic Press (1982) 51-77.
15. Nonaka, I. and Takeuchi, H: The Knowledge Creating Company, Oxford University Press, Oxford (1995).
16. Ayala, G: Intelligent Agents for a Lifelong Learning Environment in Information Technologies. International Journal of Continuing Engineering Education and Life-Long Learning (2000) (to appear).
17. Wolf, K. D: Analyzing the Process of Learning in a Web Based Community of Learners. Proceedings of the ED-TELECOM 96, World Conference on Educational Telecommunications, Boston, USA, Patricia Carlson and Filia Makedo (Eds.). AACE, (1996) 337-342.
18. Lin, X. and Hmelo, C. E: Design and Development of Technology Supported Learning Communities. Proceedings of ICCE 95, International Conference on Computers in Education 1995, Singapore, AACE, David Jonassen and Gordon McCalla (Eds.). (1995) 519 - 526.

Author Index