Computer Science & Technology Series

XVI Argentine Congress of Computer Science

Selected Papers

Computer Science & Technology Series

XVI Argentine Congress of Computer Science
Selected Papers

Guillermo Simari / Hugo Padovani
(Eds)

Editorial
de la Universidad
de La Plata

De Giusti, Armando
Computer Science & Technology Series : XVI Argentine Congress of Computer Science : selected papers . - 1a ed. - La Plata : Universidad Nacional de La Plata, 2011.
276 p. ; 24x16 cm.

ISBN 978-950-34-0757-8

1. Informática. 2. Ciencias de la Computación. I. Título.
CDD 005.3

Computer Science & Technology Series
XVI Argentine Congress of Computer Science
Selected Papers

Diseño y diagramación: Ignacio Bedatou

Editorial de la Universidad de La Plata (Edulp)
47 N.° 380 / La Plata B1900AJP / Buenos Aires, Argentina
+54 221 427 3992 / 427 4898
editorial@editorial.unlp.edu.ar
www.editorial.unlp.edu.ar

Edulp integra la Red de Editoriales Universitarias (REUN)

Primera edición, 2011
ISBN 978-950-34-0757-8
Queda hecho el depósito que marca la Ley 11.723
© 2011 – Edulp
Impreso en Argentina

Topics

XI Intelligent Agents and Systems Workshop
Chairs Guillermo Simari (UNSur) Guillermo Leguizamón (UNSL) Laura Lanzarini (UNLP)

X Distributed and Parallel Processing Workshop
Chairs Armando De Giusti (UNLP) Marcela Printista (UNSL) Jorge Ardenghi (UNSur)

IX Information Technology Applied to Education Workshop
Chairs Cristina Madoz (UNLP) María Eugenia Márquez (UNPA) Sonia Rueda (UNSur) Claudia Russo (UNNOBA)

VIII Graphic Computation, Imagery and Visualization Workshop
Chairs Silvia Castro (UNSur) Javier Giacomantone (UNLP) Roberto Guerrero (UNSL)

VII Software Engineering Workshop
Chairs Patricia Pesado (UNLP) Elsa Estévez (UNU) Alejandra Cechich (UNCOMA)

VII Database and Data Mining Workshop
Chairs Olinda Gagliardi (UNSL) Hugo Alfonso (UNLPam) Rodolfo Bertone (UNLP)

V Architecture, Nets and Operating Systems Workshop
Chairs Javier Diaz (UNLP) Antonio Castro Lechtaller (UTN) Hugo Padovani (UMorón)

II Innovation in Software Systems Workshop
Chairs Pablo Fillottrani (UNSur) Nelson Acosta (UNCPBA)

II Signal Processing and Real-Time Systems Workshop
Chairs Oscar Bría (INVAP) Horacio Villagarcia Wanza (UNLP)

Program Committee

Acosta, Nelson (Argentina)
Aguilar, José (Venezuela)
Alfonso, Hugo (Argentina)
Ardenghi, Jorge (Argentina)
Areta, Javier (Argentina)
Bertone, Rodolfo (Argentina)
Bria, Oscar (Argentina)
Castro Lechtaller Antonio (Argentina)
Castro, Silvia (Argentina)
Cechich, Alejandra (Argentina)
Collazos Ordóñez, César Alberto (Colombia)
Cukierman, Uriel (Argentina)
De Giusti, Armando (Argentina)
Díaz, Javier (Argentina)
El Saddik, Abed (Canadá)
Esquivel, Susana (Argentina)
Estevez, Elsa (United Nations)
Falappa, Marcelo (Argentina)
Fillottrani, Pablo (Argentina)
Finochietto, Jorge (Italia)
Forradellas, Raymundo (Argentina)
Gagliardi, Olinda (Argentina)
Giacomantone, Javier (Argentina)
Gröller, Eduard (Austria)
Guerrero, Roberto (Argentina)
Janowski, Tomasz (United Nations)
Lanzarini, Laura (Argentina)
Leguizamón, Guillermo (Argentina)
Luque, Emilio (España)
Madoz, Cristina (Argentina)
Malbran, María (Argentina)
Marín, Mauricio (Chile)
Marquez, María Eugenia (Argentina)
Naiouf, Marcelo (Argentina)
Obac Roda, Valentín (Brasil)
Padovani, Hugo (Argentina)
Pesado, Patricia (Argentina)
Piattini, Mario (España)
Piccoli, María F. (Argentina)
Printista, Marcela (Argentina)
Puppo, Enrico (Italia)
Ramió Aguirre, Jorge (España)
Rossi, Gustavo (Argentina)
Rueda, Sonia (Argentina)
Russo, Claudia (Argentina)
Santos, Juan Miguel (Argentina)
Sanz, Cecilia (Argentina)
Scopigno, Roberto (Italia)
Steinmetz, Ralf (Alemania)
Suppi, Remo (España)
Tarouco, Liane (Brasil)
Tirado, Francisco (España)
Velho, Luis (Brasil)
Vendrell, Eduardo (España)
Vénere, Marcelo (Argentina)
Vilanova i Bartrolí, Anna (Holanda)
Villagarcía Wanza, Horacio (Argentina)
Viola, Ivan (Noruega)
Vizcaino, Aurora (España)
Zamarro, Jose Miguel (España)

Organizing Commitee

Universidad de Morón –
Facultad de Informática,
Ciencias de la Comunicación y Técnicas Especiales – Argentina

President
Padovani, Hugo René

Coordinator
Ierache, Jorge Salvador Ierache

Collaborators
Panizzi, Marisa
Louro, Anahí
Sattolo, Iris
Malmlöf, Carolina
Preface

Preface

CACIC Congress

CACIC is an annual Congress dedicated to the promotion and advancement of all aspects of Computer Science. The major topics can be divided into the broad categories included as Workshops (Intelligent Agents and Systems, Distributed and Parallel Processing, Software Engineering, Architecture, Nets and Operating Systems, Graphic Computation, Imagery and Visualization, Information Technology applied to Education, Databases and Data Mining, Innovation in Software Systems, Theory, Real time Systems, Models and Optimization).

The objective of CACIC is to provide a forum within which to promote the development of Computer Science as an academic discipline with industrial applications, trying to extend the frontier of both the state of the art and the state of the practice.

The main audience for, and participants in, CACIC are seen as researchers in academic departments, laboratories and industrial software organizations.

CACIC started in 1995 as a Congress organized by the Network of National Universities with courses of study in Computer Science (RedUNCI), and each year it is hosted by one of these Universities. RedUNCI has a permanent Web site where its history and organization are described: http://redunci.info.unlp.edu.ar.

CACIC 2010 in Buenos Aires

CACIC'10 was the sixteenth Congress in the CACIC series. It was organized by the School of Computer Science of the University of Moron. The Congress included 10 Workshops with 104 accepted papers, 1 main Conference, 4 invited tutorials, different meetings related with Computer Science Education (Professors, PhD students, Curricula) and an International School with 5 courses. (http://www.cacic2010.edu.ar/).

CACIC 2010 was organized following the traditional Congress format, with 10 Workshops covering a diversity of dimensions of Computer Science Research. Each topic was supervised by a committee of three chairs of different Universities.
The call for papers attracted a total of 195 submissions. An average of 2.6 review reports were collected for each paper, for a grand total of 507 review reports that involved about 300 different reviewers.
A total of 104 full papers were accepted and 20 of them were selected for this book.

Acknowledgments

CACIC 2010 was made possible due to the support of many individuals and organizations. The School of Engineering of the University of Moron, RedUNCI, the Secretary of University Policies, and the National Agency of Scientific and Technological Advancement were the main institutional sponsors.
This book is a very careful selection of best qualified papers. Special thanks are due to the authors, the members of the workshop committees, and all reviewers, for their contributions to the success of this book.

Ing. Armando De Giusti
Dr. Guillermo Simari
RedUNCI

Table of Contents

XI

Intelligent Agents and Systems Workshop

An Argumentative Approach to Local-As-View Integration of Ontologies for the Semantic Web

SERGIO A. GÓMEZ[1], CARLOS I. CHESÑEVAR[1,2]
AND GUILLERMO R. SIMARI[1]

[1] Artificial Intelligence Research and Development Laboratory, Department of Computer Science and Engineering, Universidad Nacional del Sur, Av. Alem 1253, (8000) Bahía Blanca, Argentina. {sag,cic,grs}@cs.uns.edu.ar.
[2] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina.

Abstract. *Ontology integration is the problem of combining data sources in the Semantic Web. Concepts in source ontologies are captured in terms of concepts defined in a global ontology. In traditional approaches to reasoning with ontologies, when some of the ontologies involved are inconsistent, the knowledge engineer has to repair the ontologies in order to extract useful information from them. In this article we show how to perform local-as-view integration of possibly inconsistent ontologies by using Defeasible Logic Programming, thus allowing to reason automatically with ontology integration systems in the presence of inconsistency.*

1. Introduction

The *Semantic Web* (SW) [1] is a vision of the Web where resources have exact meaning assigned in terms of ontologies [2], thus enabling agents to reason about them. Ontologies in the SW are defined in the OWL language, whose underlying semantics is based on the *Description Logics* (DL) [3], for which specialized reasoners exist [4]. *Description Logic Programming* (DLP) is an alternative approach to reason with DL ontologies that proposes translating them into the language of logic programming (LP) [5]. Although DLP offers several advantages in terms of efficiency and reuse of existing LP tools (such as Prolog environments), that approach is incapable of reasoning in the presence of inconsistent ontologies. Thus we have developed a framework called *δ–ontologies* [6] for reasoning with inconsistent DL ontologies based on *Defeasible Logic Programming* (DeLP) [7].
As the World Wide Web is constituted by a variety of information sources, in order to extract information from such sources, their semantic integration and reconciliation is required [8]. Indeed, reuse of existing ontologies is often not possible without considerable effort. When one wants to reuse different ontologies together, those ontologies have to be combined in some way. This can be done by integrating the ontologies, which means that they are merged into a single new ontology, or the ontologies can be kept separate. In both cases, the ontologies have to be aligned, which means that they have to be brought into mutual agreement [9]. A particular source of inconsistency is

related to the use of imported ontologies when the knowledge engineer has no authority to correct them, and as these imported ontologies are usually developed independently, their combination could also result in inconsistencies. One kind of such integration is known as *local-as-view* (LAV) integration [8], where concepts of the local ontologies are mapped to queries over a global ontology.
In this article, we present an approach for modeling LAV ontology integration when the involved ontologies can be potentially inconsistent. The ontologies are expressed in the language of DL but we give semantics to them in terms of DeLP. The alignments between the local and global ontologies are expressed as DL inclusion axioms that are also interpreted as DeLP sentences. As the ontologies are potentially inconsistent, a dialectical analysis is performed on the interpretation of both the ontologies and the mappings from the local to the global ontology.
The rest of this paper is structured as follows. In Section 2 we present the fundamentals of Description Logics and Defeasible Logic Programming along with a brief introduction to the δ-ontologies framework for reasoning with possibly inconsistent ontologies. In Section 3, we extend δ-ontologies for performing local-as-view integration. Finally Section 4 concludes the paper.

2. Knowledge Representation and Reasoning with δ-Ontologies

2.1 Fundamentals of Description Logics and Defeasible Logic Programming

Description Logics (DL) [3] are a family of knowledge representation formalisms based on the notions of *concepts* (unary predicates, classes) and *roles* (binary relations) that allow building complex concepts and roles from atomic ones. Let *C*, *D* stand for concepts, *R* for a role and *a*, *b* for individuals. Concept descriptions are built from concept names using the constructors conjunction ($C \sqcap D$), disjunction ($C \sqcup \mathrm{D}$), complement ($\neg C$), existential restriction ($\exists R.C$), and value restriction ($\forall R.C$). To define the semantics of concept descriptions, concepts are interpreted as subsets of a domain of interest, and roles as binary relations over this domain. Further extensions are possible including inverse (P^-) and transitive (P^+) roles. A DL ontology consists of two finite and mutually disjoint sets: a *Tbox* which introduces the *terminology* and an *Abox* which contains facts about particular objects in the application domain. Tbox statements have the form $C \sqsubseteq D$ (*inclusions*) and $C \equiv D$ (*equalities*), where *C* and *D* are (possibly complex) concept descriptions. Objects in the Abox are referred to by a finite number of *individual names* and these names may be used in two types of assertional statements: *concept assertions* of the type $a{:}C$ and *role assertions* of the type $\langle a, b\rangle{:}R$.

Defeasible Logic Programming (DeLP) [7] provides a language for knowledge representation and reasoning that uses *defeasible argumentation* [10] to decide between contradictory conclusions through a *dialectical analysis*. In a DeLP program $\Pi = (\Pi, \Delta)$, a set Π of strict rules $P \leftarrow Q_1, \ldots, Q_n$, and a set Δ of defeasible rules $P \ast Q_1, \ldots, Q_n$ can be distinguished. An *argument* $\langle A, H \rangle$ is a minimal non-contradictory set of ground defeasible clauses A of Δ that allows to derive a ground literal H possibly using ground rules of Π. Since arguments may be in conflict (concept captured in terms of a logical contradiction), an attack relationship between arguments can be defined. Generalized specificity [11] is the criterion used to decide between two conflicting arguments. If the attacking argument is strictly preferred over the attacked one, then it is called a *proper defeater*. If no comparison is possible, or both arguments are equi-preferred, the attacking argument is called a *blocking defeater*. In order to determine whether a given argument A is ultimately undefeated (or *warranted*), a dialectical process is recursively carried out, where defeaters for A defeaters for these defeaters, and so on, are taken into account. The answer to a query H w.r.t. a DeLP program Π takes such dialectical analysis into account and can be one of *yes*, *no*, *undecided*, or *unknown*.

2.2 Reasoning with Inconsistent DL Ontologies in DeLP

In the presence of inconsistent ontologies, traditional DL reasoners (such as RACER [4] issue an error message and stop further processing. Thus, the burden of repairing the ontology (*i.e.*, making it consistent) is on the knowledge engineer. In a previous work [12], we showed how DeLP can be used for coping with inconsistencies in ontologies such that the task of dealing with them is automatically solved by the reasoning system. We recall some of the concepts for making this article more self-contained.

Definition 1 (δ-Ontology). Let C be an Λ_b-class, D an Λ_h-class, A, B Λ_{hb}-classes, P, Q properties, a, b individuals. Let T be a set of inclusion and equality sentences in Λ_{DL} of the form $C \sqsubseteq D$, $A \equiv B$, $\top \sqsubseteq \forall P.D$, $\top \sqsubseteq \forall P^{-}.D$, $P \sqsubseteq Q$, $P \equiv Q$, $P \equiv Q^{-}$, or $P^{+} \sqsubseteq P$ such that T can be partitioned into two disjoint sets T_S and T_D. Let A be a set of assertions disjoint with T of the form $a{:}D$ or $\langle a,b \rangle{:}P$. A *δ-ontology* Σ is a tuple (T_S, T_D, A). The set T_S is called the *strict terminology* (or Sbox), T_D the *defeasible terminology* (or Dbox) and A the *assertional box* (or Abox).

Example 1. Consider the δ-ontology $\Sigma_1 = (T_S^1, T_D^1, A^1)$ presented in Fig. 1. The strict terminology T_S^1 says that somebody who is checking mail uses a web browser. The defeasible terminology T_D^1 expresses that those who study usually pass exams, someone who is sitting at a computer is normally studying unless he is web surfing, those who do not study usually do not pass, if someone is using a web browser then he is presumably web surfing unless he is reading Javadoc documentation. The set A^1 asserts that it is known that John, Paul and Mary are sitting at a computer; Paul is using a browser; finally, Mary is also checking mail and reading Javadoc documentation. Notice that the traditional (in the sense of [3]) DL ontology ($T_S{}^1 \cup T_D{}^1$, A^1) is incoherent since somebody who both sits at a computer and web surfs then belongs both to the Studies concept and to its complement, rendering the concept empty.

Strict terminology $T_S{}^1$:	**Assertional box A^1:**
Checks_web_mail ⊑ Uses_browser	JOHN : Sits_at_computer
Defeasible terminology $T_D{}^1$:	PAUL : Sits_at_computer
Studies ⊑ Pass	PAUL : Uses_browser
Sits_at_computer ⊑ Studies	MARY : Sits_at_computer
Sits_at_computer ⊓ Web_surfing ⊑ ¬Studies	MARY : Checks_web_mail
¬Studies ⊑ ¬Pass	MARY : Reads_javadoc
Uses_browser ⊑ Web_surfing	
Uses_browser ⊓ Reads_javadoc ⊑ ¬Web_surfing	

Fig. 1. Ontology $\Sigma_1 = (T_S{}^1, T_D{}^1, A^1)$

For assigning semantics to a δ-ontology we defined two translation functions T_Δ and T_Π from DL to DeLP based on the work of [5]. The basic premise for achieving the translation of DL ontologies into DeLP is based on the observation that a DL inclusion axiom "$C \sqsubseteq D$" is regarded as a First-Order Logic statement "$(\forall x)(C(x) \rightarrow D(x))$", which in turn is regarded as a Horn-clause "$d(X) \leftarrow c(X)$".[1] Naturally "$C \sqcap D \sqsubseteq E$" is treated as "$e(X) \leftarrow c(X), d(X)$". Lloyd-Topor transformations are used to handle special cases as conjunctions in the head of rules and disjunctions in the body of rules; so "$C \sqsubseteq D \sqcap E$" is interpreted as two rules "$d(X) \leftarrow c(X)$" and "$e(X) \leftarrow c(X)$" while "$C \sqcup D \sqsubseteq E$" is transformed into "$e(X) \leftarrow c(X)$" and "$e(X) \leftarrow d(X)$". Likewise axioms of the form "$\exists r.C \sqsubseteq D$" are treated as "$d(X) \leftarrow r(X,Y), c(Y)$". Dbox axioms are treated as *defeasible* and are transformed using the T_Δ function (*e.g.*, $\mathrm{T}_\Delta(C \sqsubseteq D)$ is interpreted as "$d(X) \prec c(X)$"); Sbox axioms are considered *strict* and are transformed using T_Π (*e.g.*, $\mathrm{T}_\Pi(C \sqsubseteq D)$ is interpreted as $\{d(X) \leftarrow c(X)),(\sim c(X) \leftarrow \sim d(X))\}$).[2] Abox assertions are always considered strict (*e.g.*, T_Π $(a{:}C)$ is regarded as a fact $c(a)$ and T_Π $(\langle a,b\rangle{:}\ r)$ as $r(a,b)$). Formally:

Definition 2 (Interpretation of a δ-ontology). Let $\Sigma=(T_S,T_D,A)$ be a δ-ontology. The *interpretation of* Σ is a DeLP program $\Pi=(\mathrm{T}_\Pi(T_S) \cup \mathrm{T}_\Pi(A), \mathrm{T}_\Delta(T_D))$.
Notice that in order to keep consistency within an argument, we must enforce some internal coherence between the Abox and the Tbox; namely given a δ-ontology $\Sigma=(T_S, T_D, A)$, it must not be possible to derive two complementary

[1] Following standard logic programming notation, in DeLP rules we note constant and predicate names with an initial lowercase and variable names with an initial uppercase.
[2] The function T_Δ computes transposes of rules to allow for the application of *modus tollens*.

literals from $T_{\Pi}(T_S) \cup T_{\Pi}(A)$. We recall how we interpret the reasoning task of *instance checking* [3, p. 19] in δ-ontologies:

Definition 3 (Potential, justified and strict membership of an individual to a class). Let Σ be a δ-ontology, *C* a class name, *a* an individual, and Π the interpretation of Σ.

1. The individual *a* potentially belongs to class *C* iff there exists an argument ⟨A,*C*(*a*)⟩ w.r.t. Π;
2. the individual *a* justifiedly belongs to class *C* iff there exists a warranted argument ⟨A,*C*(*a*)⟩ w.r.t. Π, and,
3. the individual *a* strictly belongs to class *C* iff there exists an argument ⟨∅,*C*(*a*)⟩ w.r.t. Π.

Example 2 (Continues Ex. 1). Consider again the δ-ontology Σ_1, which is interpreted as the DeLP program Π_1 according to Def. 2 as shown in Fig. 2. From Π_1, we can determine that John justifiedly belongs to the concept Pass in Σ_1 as there exists a warranted argument structure ⟨A_1, *pass*(*john*)⟩ that says that John will pass the exam as he studies (because he sits at a computer), where A_1={(*pass*(*john*)∗*studies*(*john*)), (*studies*(*john*)∗*sits_at_computer*(*john*))}. We cannot reach a decision w.r.t. the membership of Paul to the concept Pass because there are two arguments attacking each other, so the answer to the query *pass*(*paul*) is *undecided*. Formally, there exist two arguments ⟨B_1,*pass*(*paul*)⟩ and ⟨B_2, ~*pass*(*paul*)⟩, where: B_1={ (*pass*(*paul*)∗*studies*(*paul*)), (*studies*(*paul*)∗*sits_at_computer*(*paul*))}, and B_2={ (~*pass*(*paul*)∗ ~*studies*(*paul*)), (~*studies*(*paul*)∗*sits_at_computer*(*paul*), *web_surfing*(*paul*)), (*web_surfing*(*paul*) ∗*uses_browser*(*paul*))}.
In the case of Mary's membership to Pass, there is an argument ⟨X_1,*pass*(*mary*)⟩ that has two defeaters, ⟨X_2, ~*pass*(*mary*)⟩ and ⟨X_3, ~*studies*(*mary*)⟩, which are both defeated by ⟨X_4, ~*web_surfing*(*mary*)⟩, where X_1={(*pass*(*studies*)∗*studies*(*mary*)), (*studies*(*mary*)∗*sits_at_ computer* (*mary*))}; X_2={(~*pass*(*mary*)∗~*studies*(*mary*))} ∪ X_3; X_3={(~*studies*(*mary*) ∗*sits_at_computer*(*mary*), *web_surfing*(*mary*)), (*web_surfing*(*mary*) ∗*uses_ browser*(*mary*))}, and X_4={(~*web_surfing*(*mary*) ∗*uses_browser*(*mary*), *reads_javadoc*(*mary*)), (*uses_browser*(*mary*) ∗*checks_web_mail*(*mary*))}. Therefore, Mary belongs justifiedly to the concept Pass as the literal *pass*(*mary*) is warranted. The dialectical trees for the three queries are depicted graphically in Fig. 3.[3]

[3] In a dialectical tree nodes are labeled as either defeated (*D*) or undefeated (*U*). Leaves are always labeled as undefeated; a node is labeled as undefeated iff all of its children are labeled as defeated, otherwise a node is labeled as defeated.

DeLP program $\Pi_1=(\Pi_1, \Delta_1)$ **obtained from** Σ_1**:**

Facts and strict rules Π_1:

sits_at_computer(*john*).
uses_browser(*paul*).
checks_web_mail(*mary*).
uses_browser(X) ← *checks_web_mail*(X).
sits_at_computer(*paul*).
uses_browser(*mary*).
reads_javadoc (*mary*).
~*checks_web_mail*(X) ← ~*uses_browser*(X).

Defeasible rules Δ_1:

pass(X) ∗ *studies*(X).
studies(X) ∗*sits_at_computer*(X).
~*studies*(X)∗ *sits_at_computer*(X), *web_surfing*(X).
~*pass*(X) ∗~*studies*(X).
web_surfing(X) ∗*uses_browser*(X).
~*web_surfing*(X) ∗ *uses_browser*(X), *reads_javadoc*(X).

Fig. 2. DeLP program Π_1 interpreting ontology Σ_1

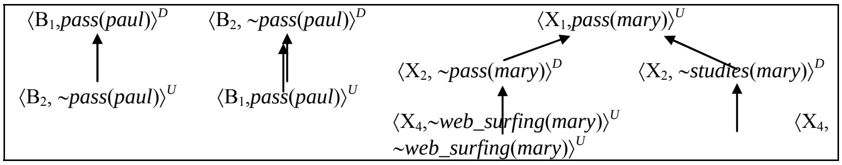


Fig. 3. Dialectical analyses for queries *pass*(*paul*) and *pass*(*mary*)

3. Local-as-View Integration of δ-Ontologies

Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of those data [13]. There are two main approaches to data integration called *global-as-view* (GAV) and *local-as-view* (LAV). In the LAV approach, we assume we have a global ontology Γ, a set Σ of local/source ontologies, and the mapping between the global and the local ontologies is given by associating to each term in the local ontologies a *view* ς_Γ over the global ontology [8, Sect. 4]. The intended meaning of associating with a term C in Σ a view ς_Γ over Γ is that such a view represents the best way to characterize the instances of C using the concepts in Γ. The correspondence between C and the associated view can be *sound* (all the individuals satisfying C satisfy ς_Γ), *complete* (if no individual other than those satisfying C satisfies ς_Γ), and/or *exact* (the set of individuals that satisfy C is exactly the set of individuals that satisfy ς_Γ).
In the GAV and LAV approaches to data integration, the queries w.r.t. the target ontology are reformulated w.r.t. the sources. Hasse & Motik [14] (referring to [13]) explain that in the GAV systems the problem is simply reduced to *unfolding* the views, since the reformulation is explicit in the mappings. In the LAV case, the problem requires more complex reasoning steps as in the case of sound mappings is not clear how to reformulate the concepts of a source ontology in terms of a global ontology. Therefore, in this work, we will restrict the case of LAV integration to complete views.

Definition 4 (Ontology integration system). An *ontology integration system* I is a triple (Γ, Σ, M) where:

- Γ is a *global ontology* expressed as a δ-ontology over an alphabet A_Γ.
- Σ is a *set of n source ontologies* $\Sigma_1, ..., \Sigma_n$ expressed as δ-ontologies over alphabets $A_{\Sigma 1}, ..., A_{\Sigma n}$, resp. Each alphabet $A_{\Sigma i}$ includes a symbol for each concept or role name of the source Σ_i, $i=1, ..., n$.
- M is a *set of n mappings* $M_1, ..., M_n$ between Γ and $\Sigma_1, ..., \Sigma_n$, resp. Each mapping M_i is constituted by a set of *assertions* of the form $q_{\Sigma i} \mid q_\Gamma$, where q_Γ and $q_{\Sigma i}$ are queries of the same arity defined over the global ontology Γ and Σ_i, $i=1, ..., n$, resp. Queries q_Γ are expressed over the alphabet A_Γ and queries $q_{\Sigma i}$ are expressed over the alphabet $A_{\Sigma i}$. The sets $M_1, ..., M_n$ are called *bridge ontologies*.

An ontology integration system will be interpreted as a DeLP program.

Definition 5 (Interpretation of an ontology integration system). Let I=(Γ,Σ,M) be an ontology integration system such that $\Sigma=\{\Sigma_1,...,\Sigma_n\}$ and $M=\{M_1,...,M_n\}$, where $\Gamma=(T_S^\Gamma, T_D^\Gamma, A^\Gamma)$; $\Sigma=(T_S^{\Sigma i}, T_D^{\Sigma i}, A_i^{\Sigma i})$, and, $M_i=(T_S^{Mi}, T_D^{Mi})$, with $i=1,...,n$. The system I is interpreted as the DeLP program $I_{DeLP}=(\Pi,\Delta)$, with:

$$\Pi = T_\Pi(T_S^\Gamma) \cup T_\Pi(A^\Gamma) \cup (\cup_{i=1,...,n} T_\Pi(T_S^{\Sigma i})) \cup (\cup_{i=1,...,n} T_\Pi(T_S^{Mi})), \text{ and}$$
$$\Delta = T_\Delta(T_D^\Gamma) \cup (\cup_{i=1,...,n} T_\Delta(T_D^{\Sigma i})) \cup (\cup_{i=1,...,n} T_\Delta(T_D^{Mi})),$$

Possible inferences in the integrated ontology I_{DeLP} are modeled by means of a dialectical analysis in the DeLP program that is obtained when each DL sentence of the ontology is mapped into DeLP clauses. Thus conclusions supported by warranted arguments will be the valid consequences that will be obtained from the original ontology, provided the strict information in I_{DeLP} is consistent. Formally:

Definition 6 (Potential, justified and strict membership of individuals to concepts in ontology integration systems). Let I=(Γ,Σ,M) be an ontology integration system.
Let *a* be an individual name, and *C* a concept name defined in Γ.

1. Individual *a* is a *potential member* of *C* iff there exists an argument A for the literal *C*(*a*) w.r.t. DeLP program I_{DeLP}.
2. Individual *a* is a *justified member* of *C* iff there exists a warranted argument A for the literal *C*(*a*) w.r.t. DeLP program I_{DeLP}.
3. Individual *a* is an *strict member* of *C* iff there exists an empty argument for the literal *C*(*a*) w.r.t. DeLP program I_{DeLP}.

We will illustrate the above notions with an example. Notice that we label a concept *C* with the name of the ontology Σ_i to which it belongs (as in Σ_i:*C*) following the XML name-space convention.

Example3. Let us consider the problem of assigning reviewers for papers. In Fig. 4, we present a global ontology Γ_3 interpreted as: a professor with a postgraduate degree can be a reviewer; someone should not be a reviewer unless they are either a professor or have a graduate degree; however, a professor, despite not having a postgraduate degree, is accepted as a reviewer if he is an outstanding researcher. We also present local ontologies Λ_1 and Λ_2. Λ_1 expresses that John is a professor who has a PhD, Paul is also a professor but has neither a PhD nor a MSc, Mary just has a MSc, and Steve is not a professor but has a MSc. The mapping $M_{\Lambda 1,\Gamma}$ expresses that the terms MSc and PhD from local ontology Λ_1 are contained in the term postgraduated in the global ontology, and that someone have neither a MSc nor a PhD is not a postgraduate. Ontology Λ_2 expresses that *a* is an article, *b* a book, *c* a chapter and that Paul has published *a*, *b* and *c*. The mapping $M_{\Lambda 2,\Gamma}$ expresses that the view corresponding to the individuals who have published an article, a chapter and a book corresponds to the set of outstanding researchers.
The interpretation of above ontologies in DeLP yields the code presented in Fig 5. We show next the dialectical analyses that have to be performed to compute the justified membership of John, Paul, Mary and Steve to the concept Reviewer w.r.t. the ontology integration system (Γ, $\{\Lambda_1, \Lambda_2\}$, $\{M_{\Lambda 1,\Gamma}, M_{\Lambda 2,\Gamma}\}$).
The individual John is a justified member of the concept Reviewer because the argument ⟨A, *reviewer*(*john*)⟩ has no defeaters and is thus warranted (see Fig. 6(a)), with A = {(reviewer(john)∗*postgrad*(*john*), *prof*(*john*)), (*postgrad*(*john*)∗ *phd*(*john*))}.
In Paul's case, we conclude that he is a possible reviewer as he is also a justified member of the concept Reviewer. Notice that Paul is a potential member of the concept ¬Reviewer as there is an argument ⟨B_1,*~reviewer*(*paul*)⟩, with B_1={ (*~reviewer*(*paul*) ∗*~postgrad*(*paul*)), (*~postgrad*(*paul*) ∗ *~msc*(*paul*), *~phd*(*paul*))}.
However, we see that there is another argument ⟨B_2, *reviewer*(*paul*)⟩ where B_2 = {(*reviewer*(*paul*)∗*prof*(*paul*), *~postgrad*(*paul*), *outstanding*(*paul*)), (*outstanding*(*paul*)∗*published*(*paul*,*a*), *article*(*a*), *published*(*paul*,*b*), *book*(*b*), *published*(*paul*,*c*), *chapter*(*c*)), (*~postgrad*(*paul*)∗ *~msc*(*paul*), *~phd*(*paul*))}.
As B_2 is undefeated, we conclude that the literal *reviewer*(*paul*) is warranted (see Fig. 6.(b)-(c)).
Steve is not a reviewer as he is a justified member of the concept ¬Reviewer (see Fig. 6.(d)). In this case, there exists a unique (undefeated) argument ⟨X, *~reviewer*(*steve*)⟩. On the other hand, it is not possible to assess Mary's membership to the concept Reviewer as no arguments for *reviewer*(*mary*) nor *~reviewer*(*mary*) can be built.

Global ontology $\Gamma_3 = (\varnothing, T_D^{\Gamma}, \varnothing)$**:**
Prof ⊓ Postgrad ⊑ Reviewer
¬Postgrad ⊓ ¬Prof ⊑ ¬Reviewer
Prof ⊓ ¬Postgrad ⊓ Outstanding ⊑ Reviewer

Local ontology $\Lambda_1 = (\varnothing, \varnothing, A^{\Lambda 1})$**:**
JOHN : Prof; JOHN : Phd; PAUL : Prof; PAUL : ¬Msc;
PAUL : ¬Msc; MARY : Msc; STEVE : ¬Prof; STEVE : Msc

Mapping $M_{\Lambda 1,\Gamma} = (\varnothing, \varnothing, A^{\Lambda 1})$ **between** Λ_1 **and** Γ**:**
Λ_1 : Msc ⊓ Λ_1 : Phd ⊑ Γ : Postgrad
Λ_1 : ¬Msc ⊓ Λ_1 : ¬Phd ⊑ Γ : ¬Postgrad

Local ontology $\Lambda_2 = (\varnothing, \varnothing, A^{\Lambda 2})$**:**
a : Article; b : Book; c : Book
⟨PAUL, a⟩ : published; ⟨PAUL, b⟩ : published; ⟨PAUL, c⟩ : published;

Mapping $M_{\Lambda 2,\Gamma} = (\varnothing, \varnothing, A^{\Lambda 2})$ **between** Λ_2 **and** Γ**:**
Λ_2 : (∃published.Article ⊓ ∃published.Book ⊓ ∃published.Chapter) ⊑ Γ : Outstanding

Fig. 4. LAV ontology integration system

4. Conclusions

We have presented an approach for performing local-as-view integration of Description Logic ontologies when these ontologies can be potentially inconsistent. We have adapted the notion of ontology integration system of [8] for making it suitable for the δ-ontology framework, presenting both formal definitions and a case study. We offer several advantages over previous efforts (such as [8,14]) as our proposal is capable of dealing with inconsistent ontologies. This work also presents a difference with previous works of ours (such as [12,15]) as those works focused on the problem of global-as-view integration of ontologies. Despite this advancement, the proposed approach is only useful in the case of complete mappings, and therefore the case for local-as-view integration with sound and exact mappings remains as an open problem and is part of our current research efforts.

DeLP program $(\varnothing, \Delta_\Gamma)$ **obtained from** Γ_3**:**
reviewer(*X*) ∗ *postgrad*(*X*), *prof*(*X*). ~*reviewer*(*X*) ∗ ~*postgrad*(*X*).
~*reviewer*(*X*) ∗ ~*prof*(*X*). *reviewer*(*X*) ∗ *prof*(*X*), ~*postgrad*(*X*), *outstanding*(*X*).
DeLP program $(\Pi^{\Lambda 1}, \varnothing)$ **obtained from** Λ_1**:**
prof(*john*). *prof*(*john*). *prof*(*paul*). ~*msc*(*paul*).
~*phd*(*paul*). *msc*(*mary*). ~*prof*(*paul*). *msc*(*steve*).
Mapping $M_{\Lambda 1,\Gamma}$ **expressed in DeLP as** $\Delta_{\Lambda 1\Gamma}$**:**
Γ : *postgrad*(*X*) ∗ Λ_1 : *msc*(*X*). Γ : *postgrad*(*X*) ∗ Λ_1 : *phd*(*X*).
Γ : ~*postgrad*(*X*) ∗ Λ_1 : ~*msc*(*X*), Λ_1 : ~*phd*(*X*).
DeLP program $(\Pi^{\Lambda 2}, \varnothing)$ **obtained from** Λ_2**:**
article(*a*). *book*(*b*). *chapter*(*c*).
published(*paul*, *a*). *published*(*paul*, *b*). *published*(*paul*, *c*).
Mapping $M_{\Lambda 2,\Gamma}$ **expressed in DeLP as** $\Delta_{\Lambda 2\Gamma}$**:**
Γ : *outstanding*(*X*) ∗ Λ_2 : *published*(*X*,*Y*), Λ_2 : *article*(*Y*), Λ_2 : *published*(*X*,*Z*), Λ_2 : *article*(*Z*),
Λ_2 : *published*(*X*,*W*), Λ_2 : *article*(*W*).

Fig. 5. Ontologies Γ_3, Λ_1 and Λ_2 expressed in DeLP

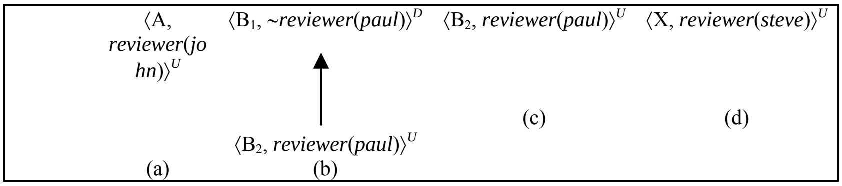


Fig. 6. Dialectical trees for *reviewer*(*john*), *reviewer*(*paul*), *reviewer*(*steve*)

Acknowledgements. This research is funded by Projects PIP 112-200801-02798 (CONICET, Argentina), PGI 24/ZN10, PGI 24/N006 (SGCyT, UNS, Argentina) and Universidad Nacional del Sur.

References

1. Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web. Scient. American.
2. Gruber, T.R. (1993). A translation approach to portable ontologies. Knowledge Acquisition **5**(2), 199-220.
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds. (2003). The Description Logic Handbook - Theory, Implementation and Applications. Cambridge University Press.
4. Haarslev, V., Möller, R. (2001). RACER System Description. Technical report, University of Hamburg, Computer Science Department.
5. Grosof, B.N., Horrocks, I., Volz, R., Decker, S. (2003). Description Logic Programs: Combining Logic Programs with Description Logics. WWW2003, May 20-24, Budapest, Hungary.
6. Gómez, S.A. (2009) Integración de Argumentación Rebatible y Ontologí-as en el Contexto de la Web Semántica: Formalización y Aplicaciones. PhD thesis, Universidad Nacional del Sur.
7. García, A., Simari, G. (2004) Defeasible Logic Programming an Argumentative Approach. Theory and Practice of Logic Programmming 4(1), 95-138.
8. Calvanese, D., Giacomo, G.D., Lenzerini, M. (2001). A framework for ontology integration. In: First Semantic Web Working Symposium. 303-316.
9. Klein, M. (2001). Combining and relating ontologies: an analysis of problems and solutions. In Gomez-Perez, A., Gruninger, M., Stuckenschmidt, H., Uschold, M., eds.: Workshop on Ontologies and Information Sharing, IJCAI'01, Seattle, USA, August 4-5.
10. Chesñevar, C.I., Maguitman, A., Loui, R. (2000): Logical Models of Argument. ACM Computing Surveys 32(4), December, 337-383.
11. Stolzenburg, F., García, A., Chesñevar, C., Simari, G. (2003). Computing Generalized Specificity. J. of N.Classical Logics 13(1), 87-113.
12. Gómez, S.A., Chesñevar, C.I., Simari, G.R. (2008). An Argumentative Approach to Reasoning with Inconsistent Ontologies. In Meyer, T.,

Orgun, M.A., eds.: Proc. of the Knowledge Representation in Ontologies Workshop (KROW 2008). Volume CPRIT 90., Sydney, Australia, 11-20.

13. Lenzerini, M. (2002). Data integration: A theoretical perspective. Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2002, Madison, Winsconsin, USA.
14. Haase, P., Motik, B. (2005). A mapping system for the integration of OWL-DL ontologies. In Hahn, A., Abels, S., Haak, L., eds.: IHIS 05: Proceedings of the first international workshop on Interoperability of heterogeneous information systems, ACM Press, NOV, 9-16.
15. Gómez, S.A., Chesñevar, C.I., Simari, G.R. (2010). Reasoning with Inconsistent Ontologies Through Argumentation. Applied Artificial Intelligence 24(1), 102-148.

Parallel ACO algorithms for 2D Strip Packing

CAROLINA SALTO[1], GUILLERMO LEGUIZAMÓN[2] AND ENRIQUE ALBA[3]

[1] LISI - Universidad Nacional de La Pampa, Calle 110 esq. 9,
Gral. Pico, La Pampa, Argentina. saltoc@ing.unlpam.edu.ar.
[2] LIDIC - Universidad Nacional de San Luis, Ejército de los Andes 950, San Luis, Argentina.
legui@unsl.edu.ar.
[3] GISUM - Universidad de Málaga, Campus de Teatinos, 29071, Málaga, España.
eat@lcc.uma.es.

Abstract. *In this paper we present a study of a parallel Ant Colony System (ACS) for the two-dimensional strip packing problem. In our computational study, we emphasize the influence of the incorporation of the received information in the target subcolony. Colonies send their best solutions instead of sending information from the pheromone matrix, as happens in traditional parallel ACS. The solution arriving to a colony can provide further exploitation around promising solutions as this arrived solution can be used in both, the local update of the pheromone trail and the construction solution process of an ant. The aim of the paper is to report experimental results on the behavior of different types of parallel ACS algorithms, regarding solution quality and parallel performance.*

1. Introduction

All parallel ACO options studied so far in the literature have a common characteristic: the construction of a single solution by an ant is not split between multiple processors [8, 13]. The reason is that the solution construction process in ACO is typically a sequential process which is difficult to split in several independent parts. Consequently, the minimum grain size of parallel ACO is the construction of a single solution.
The majority of the parallel ACO algorithms assign more than one ant on each processor [2, 12]. When several ants are placed on a single processor and these ants work more closely than those ants in other processors, this group of ants are often called a *colony*. Those ACO algorithms that have several colonies of ants using their own pheromone matrix and where the pheromone matrices of different colonies are not necessarily equal are called *multicolony ACO algorithms*. This type of ACO algorithms are used in this work [8, 12]. Multicolony ACO algorithms are well suited for parallelization because a processor can host a colony of ants [1].
In this work we evaluate the application of three parallel ACO algorithms to solve a strip packing problem. We considered three parallel strategies to implement a parallel ant colony. In one of them, no communication is required between subcolonies. The other two strategies exchange information between the subcolonies. Traditional implementations exchange information

of the pheromone matrix, but in this work the subcolonies send the best solution found-so-far to their neighbors [10]. The difference in these last two strategies is the way in which the information arriving to the target subcolonies is used. All algorithms are studied from the numerical point of view but also from the parallel performance.
The article is organized as follows. Section 2 contains an explanation of the 2SPP. Section 3 describes the multicolony ACS used to solve the 2SPP. In Section 4, we explain the parameter settings of the algorithms used in the experimentation. Section 5 reports on the performance of the algorithms studied and finally, in Section 6 we give some conclusions and analyze future lines of research.

2. The 2D Strip Packing Problem

Packing problems involve the construction of an arrangement of pieces that minimize the total space required for that arrangement. In this paper, we specifically consider the two-dimensional Strip Packing Problem (2SPP), which consists of a set of M rectangular pieces, each one defined by a width $w_i \leq W$ and a height h_i, (i=1...M). The goal is to pack the pieces in a larger rectangle, the *strip*, with a fixed width W and unlimited length, minimizing the required strip length; an important restriction is that the pieces have to be packed with their sides parallel to the sides of the strip, without overlapping.
In the present study some additional constrains are imposed: pieces must not be rotated and they have to be packed into three-stage level packing patterns. In these patterns, pieces are packed by horizontal levels (parallel to the bottom of the strip). Inside each level, pieces are packed bottom left justified and, when there is enough room in the level, pieces with same width are stacked one above the other. Three-stage level patterns are used in many real applications in the glass, wood, and metal industries, and this is the reason for incorporating this restriction in the problem. The 2SPP is representative of a wide class of combinatorial problems, being a NP-hard [9] one.

3. Parallel Ant Colony System to the 2SPP

Ant Colony System (ACS) [7,6] is one of the most representative algorithms derived from the Ant Colony Optimization (ACO) metaheuristic to deal with combinatorial optimization and other problems. It uses a colony of artificial ants which stochastically build new solutions using a combination of heuristic information and artificial pheromone trail. This pheromone trail is reinforced according to the quality of the solutions built by the ants.
Like many other metaheuristic approaches, the ACO metaheuristic admits direct parallelization schemes. Randall and Lewis [15] proposed an interesting classification of the parallelization strategies for ACO metaheuristic: parallel independent ant colonies, parallel interacting ant

colonies, parallel ants, parallel evaluation of solution elements, and a combination of two of the mentioned strategies. In this work, we considered a version of the first and second strategies, which are described in the following paragraphs. The reason for this election is based on the good performance observed in the solution of other problems [2], and they performed similarly regarding the qualities of the results obtained.
In the case of parallel independent ant colony, there is a number of sequential ACSs which are put on different processors. This method, called *dACS_{ni}*, has the particularity that colonies do not send information. This alternative has a positive effect over the behavior since each colony, which is running in a different processor, can specialize in different regions of the search space. The parallel interacting ant colonies strategies is similar to the previous one, except that an exchange of information between subcolonies occurs at a prefix iterations. The exchange of information is frequently associated to share the pheromone trail structure of the best performing colony among all the subcolonies. Also it is possible to send the best solutions found in each colony.

```
init_pheromoneValues($\tau$);                 \\Initialize the pheromone trails
$s_{bs}$=build_solution($\tau,\eta$);
while(not(stop condition))
    for($k \leftarrow$1to $\mu$}
        ant$_k$=build_solution$\tau,\eta$);          \\Ant ant$_k$ builds a solution
        localUpdate($\tau$, ant$_k$);               \\Local update of pheromone trials (ACS)
        apply_localSearch(ant$_k$ );
    if(exchange_iteration)
        send/receive_solution(ant,dACS$j$); \\ interaction with the neighborhood
    feromone_evaporation($\tau$)                 \\ evaporation
    for($k \leftarrow$ 1 to $\mu$)
        if($f$(ant$_k$) < $f$($s_{bs}$))
            $s_{bs}$=ant$_k$                  \\actualize the best solution, daemon activity
    globalUpdate($\tau$,ant, $s_{bs}$)        \\intensification of pheromone trails
return best solution found $s_{bs}$
```

Algorithm 1: Algorithm dACS

The distributed ACS algorithm that we use in this work is shown in Algorithm 1. The algorithm begins with the initialization of the pheromone trail associated with each transition. The principal loop of the algorithm consists of the following steps. A colony of μ ants incrementally build solutions (packing patterns) to the 2SPP applying a stochastic local decision policy that makes use of pheromone trails and heuristic information. While a solution is being built, the ant deposits a pheromone trail on the used components or connections (local updating rule). This pheromone

information will direct the search of future ants. Once the solution is created a local search procedure is activated. At preset iterations, the subcolony exchanges information with the neighborhood (the communication structure used in this work corresponds to an unidirectional ring topology). After this communication step, a pheromone evaporation is triggered, which is the process by means of which the pheromone deposited by previous ants decreases over time. The best solution s_{bs} should be updated if an ant at the current iteration found a better packing pattern. Finally, a global pheromone update is carried out in order to deposit extra pheromone to good packing pattern. The algorithm returns the best solution found-so-far.
One of the aspects to consider in the moment of designing a dACS is the information exchanged between the colonies. One choice is to send solutions that have been found in a colony to its neighbor. Another choice is to send information from the pheromone matrix. As the results of [10] indicate that the exchange of pheromone matrices is not desirable, in this work we have chosen to send the best solution found by the colony.
According to the unidirectional ring topology adopted in this work to communicate the different subcolonies, subcolony *i* influences the levels of pheromone trails of subcolony (*i*+1) mod *n* (where *n* is the number of subcolonies) by sending their best-so-far solution. In this work, the incoming solution is used in two different moments:

- *for local updating*. The solution arriving to a target subcolony is used in the local updating process together with the local set of solutions. Therefore, when only the best solution in a subcolony is allowed to update the pheromone values, the incoming solution influences the pheromone levels only when it is better than all the solutions found by ants in that iteration. This update made a very indirect use of the received information. This algorithm is referred as *dACS* in the following.
- *for a more direct use of the information*. One way to complement the use of the received information from the neighboring colony is to use the information of the arrangement of the pieces of that solution in a more direct way. Therefore, we choose another alternative which consists in to extract the good levels of the incoming solution (those levels with a less waste) and to copy them to a pseudo-solution. This pseudo-solution is incomplete: some pieces are missed. The incoming solution is used in the local updating process as explained in previous paragraph and the process continues traditionally. In the next iteration of the algorithm, the ants begin the process of building their solution by copying the levels from the pseudo-solution and then they repeatedly apply a state transition rule to complete the packing pattern, using the pheromone trail and heuristic information. This combination allows a mix of exploitation (for the incoming solution) and exploration (for the experience of the target colony) through the respective

pheromone matrix. The algorithm implementing these ideas is called *dACS*$_{mem}$.

The following paragraphs detail how the ACS can be applied to the 2SPP [16]. This description includes the most important elements of the ACS, namely the use of heuristic information, the pheromone trail definition, the state transition rule, and the local search procedure used in order to improve the solution quality.
We maintain solutions in the form of permutations of the set of pieces [5], which will be directly translated into the corresponding packing pattern by a layout algorithm. In order to generate a 3-stage level pattern, i.e., the pieces layout, we adopt a modified *next-fit decreasing height* heuristic (NFDH) -in the following referred as *modified next-fit*, or *MNF*- which was proven to be very efficient in [14, 18]. A more in-depth explanation of the MNF procedure can be found in [18].
The objective value of a solution *s* of ant$_s$ is defined as the strip length needed to build the corresponding packing pattern. An important consideration is that two packing patterns could have the same length -so their objective values will be equal- however, from the point of view of reusing the trim loss, one of them can be actually better because the trim loss in the last level (which still connects with the remainder of the strip) is greater than the one present in the last level in the other layout. Therefore we use the following objective function:

$$f(s) = strip.length - \frac{l.waste}{strip.lenght \times W} \quad (1)$$

where *strip.length* is the length of the packing pattern corresponding to the permutation *s* and *l.waste* is the area of reusable trim loss in the last level *l* of the packing pattern. Hence, function *f* is both simple and accurate.

3.1 Heuristic definition

For the 2SPP, the problem-dependent heuristic information used is the height of piece *j*, i.e., the heuristic value for a piece *j* is $\eta_j = h_j$.

3.2 Pheromone definition

Trail τ_{ij} encodes the desirability of having a piece *i* and *j* in the same level [11]. The pheromone matrix has *M* rows and *M* columns (in a first stage, each piece is assigned to a different level, in that way initially we have *M* different levels).

3.3 Pheromone update

Once all ants have completed their packing patterns, a global pheromone updating rule is applied. In this case, only the best ant (which the respective solution is s_{bs}) is allowed to place pheromone after each iteration. This is done according to $\tau_{ij}=\rho \times \tau_{ij} + 1/f(s_{bs})$, where $0<\rho<1$ is the pheromone decay parameter, and $f(s_{bs})$ is the objective value of s_{bs}. Using only the best ant for updating makes the search much more aggressive. Global updating is intended to provide a greater amount of pheromone to good packing patterns. Moreover, while ants construct a solution, a local pheromone updating rule is applied, where the effect is to make the desirability of edges change dynamically. The local updating is made according to the following expression: $\tau_{ij} =(1-\xi) \times \tau_{ij} + \xi \times \Delta\tau_{ij}$, where $0<\xi<1$ is a parameter and $\Delta\tau_{ij}$ is set as τ_{min}. Dorigo and Gambardella [7] used this expression to run their experiments with good results.
Another way to promote exploration is by defining a lower limit (τ_{min}) for the pheromone values. The following formula sets the value of τ_{min} [11] as:

$$\tau_{\min} = \frac{\frac{1}{1-\rho}(1-\sqrt[M]{pbest})}{(avg-1)\sqrt[M]{pbest}} \qquad (2)$$

where *pbest* is the approximation of the real probability to construct the best solution, *avg* is the average number of pieces to choose from at every decision point when building a solution, defined as $M/2$. Also an evaporation phase occurs at each iteration by updating the pheromone trail by $\tau_{ij} = \gamma \times \tau_{ij}$, where $0<\gamma<1$ is a parameter.

3.4 State transition rule definition

It gives the probability with which ant *k* will choose a piece *j* as the next piece for its current level *l* in the partial solution *s*, which is given by [11]:

$$j = \begin{cases} \max_{j \in J_k(s,l)} \left[\tau_i^*(l)\right] \times \left[\eta_j\right]^\beta & \text{if } q \le q_0 \\ S & \text{otherwise} \end{cases} \qquad (3)$$

where τ_{lj} is the pheromone value for piece *j* in level *l*, η_j is the heuristic information guiding the ant, β is a parameter which determines the relative importance of pheromone information versus heuristic information, *q* is a random number uniformly distributed in [0..1], q_0 is a constant parameter ($0< q_0<1$) which determines the relative importance of exploitation versus exploration, and *S* is a random variable selected according to the probability distribution given in Equation 4.

$$p_k(s,l,j) = \begin{cases} \dfrac{\left[\tau_j^*(l)\right] \times \left[\eta_j\right]^\beta}{\sum_{g \in J_k(s,l)} \left[\tau_g^*(l)\right] \times \left[\eta_g\right]^\beta} & \text{if } j \in J_k(s,l) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

In Equations 3 and 4, $J_k(s,l)$ is the set of pieces that qualify for inclusion in the current level by ant *k*. The set includes those pieces that are still left after partial solution *s* is formed, and are light enough to fit in level *l*. The pheromone value $\tau^*_j(l)$ for a piece *j* in a level *l* is given by:

$$\tau_j^*(l) = \begin{cases} \dfrac{\sum_{i \in A_l} \tau_{ij}}{|A_l|} & \text{if } A_l \neq J_k(s,l) \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

where A_l is the set of current pieces allocated in level *l*. In other words, $\tau^*_j(l)$ is the sum of all the pheromone values of pieces already in level *l*, divided by the number of pieces in that level. This approach is similar to the one followed by Levine and Ducatelle [11].

3.5 The local search procedure

It starts from a solution created by the ACS towards to the nearest local optimum for that solution, with the aim of improving the trim loss of all levels. After this improvement phase, the pheromone trail is updated. The local search procedure used in this work consists of the application of a modified version of first-fit decreasing heuristic (FFDH), called MFF. A more in-depth explanation of the MFF procedure can be found in [18].

4. Implementation

Now we will comment on the actual implementation of the multicolony algorithms to solve the 2SPP: *i*) dACS with independent non-interacting colonies ($dACS_{ni}$) and two dACSs with colonies exchanging information: *ii*) *dACS*, which add the received solution to the solution set of the colony and *iii*) $dACS_{mem}$, which adds the received solution to the solution set as well as selects the best levels of the incoming solution with the objective that those levels will be used for ants in the next iterations.
The number of ants is set to 64, each subcolony has 64/*n* ants, where *n* represents the number of subcolonies. Each ant begins the building process of their solution with a piece randomly selected. The parameter values are the

following: β = 2, q_0=0.9, ρ=0.8, γ=0.96, and ξ=0.1. The initial pheromone value is set to τ_{min}. These parameters were used with success in [17]. For the models involving communication between subcolonies, solutions were sent with a frequency of 100 iterations following an asynchronous approach. Local search is applied to all solutions generated by the ants.
The algorithms were implemented inside MALLBA [3], a C++ software library fostering rapid prototyping of hybrid and parallel algorithms. The platform was a cluster of 16 PCs with Intel Pentium 4 at 2.4 GHz and 1GB RAM under SuSE Linux with 2.4.19-4 kernel version, and interconnected by a Fast-Ethernet at 100 Mbps.
We have considered five randomly generated problem instances with *M* equal to 100, 150, 200, 250, and 300 pieces and a known global optimum equal to 200 (the minimum length of the strip). These instances belong to the subtype of level packing patterns but the optimum value does not correspond to a 3-stage guillotine pattern. They were obtained by an own implementation of a data set generator, following the ideas proposed in [19] with the length-to-width ratio of all *M* rectangles in the range $1/3 \leq l/w \leq 3$. These instances are publicly available at http://mdk.ing.unlpam.edu.ar/~lisi/documentos/datos2spp.zip.

5. Computational Analysis

In this section we summarize the results of applying the multicolony algorithms solve the 2SPP with restrictions, using different strategies to incorporate the information of the received solution. In a first place, a comparison of the multicolony algorithms is presented by establishing the same effort, a prefixed number of iterations. After that, the view point is changed in order to review the parallel performance. Our aim is to offer meaningful results and check them from a statistical point of view. For each algorithm we have performed 30 independent runs per instance using the parameter values described in the previous section.
In order to obtain meaningful conclusions, we have performed an analysis of variance of the results. When the results followed a normal distribution, we used the t-test for the two-group case, and the ANOVA test to compare differences among three or more groups (multiple comparison test). We have considered a level of significance of α=0.05, in order to indicate a 95% confidence level in the results. When the results did not follow a normal distribution, we used the non-parametric Kruskal Wallis test (multiple comparison test), to distinguish meaningful differences among the means of the results for each algorithm.

Table 1. Best fitness values for *ACSseq* and the proposed *Dacs*

Inst	*ACSseq*		*dACS$_{ni}$*		*dACS*		*dACS$_{mem}$*	
	best	*avg±σ*	*best*	*avg±σ*	*Best*	*avg±σ*	*best*	*avg±σ*
100	215.78	$218.29_{\pm 0.88}$	215.64	$218.29\pm_{0.86}$	**214.73**	$217.93_{\pm 1.08}$	215.64	$218.18_{\pm 0.98}$
150	216.38	$217.82_{\pm 0.79}$	215.64	$218.29\pm_{0.86}$	215.69	$217.82_{\pm 0.70}$	**214.79**	$217.69_{\pm 0.88}$
200	211.61	$214.37_{\pm 1.12}$	215.64	$218.29\pm_{0.86}$	210.77	$213.29_{\pm 1.28}$	**210.68**	$213.68_{\pm 1.18}$
250	207.68	$209.20_{\pm 0.76}$	215.64	$218.29\pm_{0.86}$	207.70	$209.28_{\pm 0.74}$	**207.54**	$209.20_{\pm 0.62}$
300	213.66	$214.74_{\pm 0.57}$	215.64	$218.29\pm_{0.86}$	**211.27**	$213.98_{\pm 0.96}$	211.79	$213.92_{\pm 0.68}$

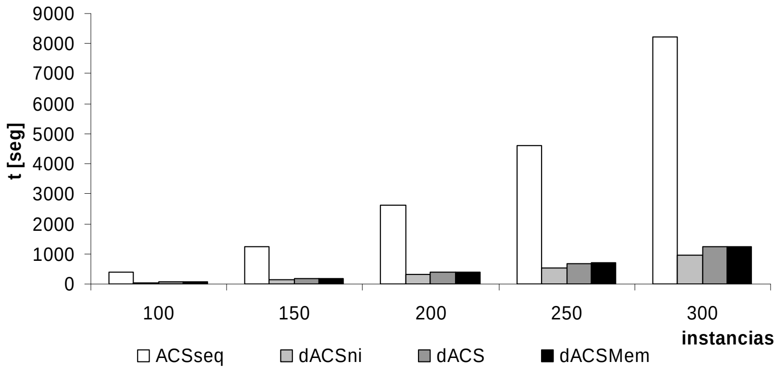


Fig. 1. Execution time for *seqACS* and each *dACS*

5.1 Results with predefined effort

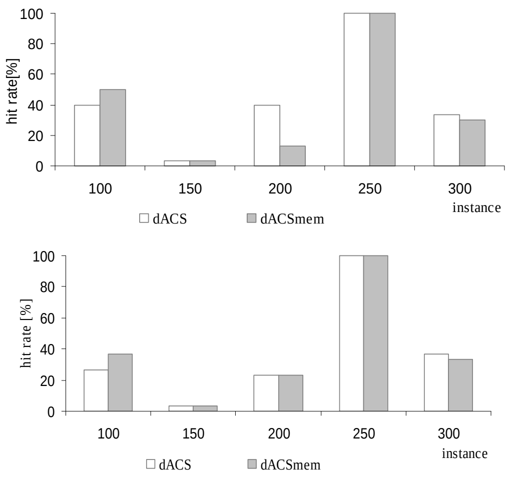
In this section, a sequential ACS, so-called *ACSseq*, is also included in the study, in order to show that the multicolony ACSs present not only lower runtimes but also better solutions to the problem. In order to make a fair comparison, all proposals stop after 65.536 evaluations (2^{32}). Table 1 shows the results of the different ACSs for each instance. The columns in this table stand respectively for the best objective value obtained (*best*) and the average objective values of the best found feasible solutions along with their standard deviations (*avg±σ*). The minimum *best* values are printed in bold.

From this table we can observe that multicolony approaches are the algorithms that reach the best packing patterns for the whole set of instances; but there are no significant differences in the statistics analysis performed for instances with *M*=100, *M*=150 and *M*=250, i.e., the statistical tests indicate that all algorithms have presented similar mean values. The more important difference between the multicolony approaches and the sequential ACS is the run time, as to be expected (see Figure 1).

There are no differences between the multicolony approaches regarding the quality of the solution found, although a small advantage in favor of *dACS$_{mem}$* is

Table 2. Quality of the target solutions to stop the algorithms

M	**100**	**150**	**200**	**250**	**300**
target	217.99	216.23	213.12	213.59	213.90



(a) (b)

Fig. 2. Hit rates (in percentage) for each *dACS* algorithm: (a) 1 processor and (b) 8 processors

observed, since it solves more effectively three of the five instances. Regarding run times, *dACS* and *dACS*$_{mem}$ present significant statistical differences only for instances with *M*=100 and *M*=250. From the examination of the mean run times values for those instances, it is observed that the differences are negligible, for example, *dACS* took about 58.94 sec. in the search, meanwhile *dACS*$_{mem}$ took about 59.76 sec. in the instance with *M*=100, meaning a difference of 0.82 sec.; similar situation is present in the instance with *M*=250. This means that the additional processing incurred in saving the received solution and in extracting their good level do not substantially affect the run time, which transforms this option in a viable alternative to obtain good solutions to 2SPP.

5.2 Results with predefined quality of solutions

Now we change the kind of analysis performed. We want to measure the time to find equivalent solutions with the dACSs proposed, in order to show their parallel characteristics. Thus we define our goal as reaching the fitness values which are shown in Table 2. To carry out this experimentation, the eight subcolonies of each dACS are put on a same processor and then every subcolony is put in a dedicated processor.

Up to now, we have presented the average results over 30 independent runs. This time, we show the hit rate of the distributed ACSs, which is presented in Figure 2. This measure is the relation between the number of execution that reached the target fitness and the total number of performed tests. It is important to highlight

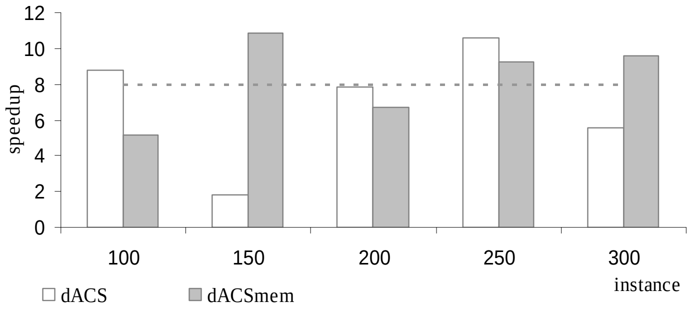


Fig. 3. Speedup values

that both *dACS* and *dACS*$_{mem}$ reach the target value in all runs in instance with *M*=250, independently of the number of processors used. The eight sub-colonies running in sequence, i.e., using only one processor, obtain a similar hit rate that the parallel approaches, in instances with *M*=150 and *M*=300. Instance with *M*=150 has been difficult for any of the dACS algorithms, since the hit rate is equal to 3.3%, meaning that in only one run from the 30 the solution obtained was better than the target.
From the point of view of the parallel performance of the algorithms, Figure 3 shows the speedup values, which are obtained following the orthodox definition of speedup [4]. We can see high speedup values in the majority of the dACS approaches, except for *dACS* and the instance with *M*=150, where the speedup value is lower than two. In particular, superlinear speedups are observed in both algorithms in instance with *M*=250. These results suggest that we are using good parallel implementations of the algorithms.

6. Conclusions

In this paper we have presented different parallel ACSs to solve the 2SPP with additional constrains. The parallelization strategy consisted in a multicolony model, where sporadic exchange of solutions between subcolonies occurs. Therefore, the exchange of solutions between subcolonies can be considered as a class of interaction among parallel ant colonies. The characteristics of the distributed models have proven to be good techniques to obtain good packing patterns, which represents a great step forward in this field.

Computational results of the three considered multicolony strategies are similar than those obtained with a sequential ACS, but $dACS_{mem}$ and *dACS* obtained the best packing patterns. The most important difference between the sequential ACS and the multicolony ACSs was observed in the run times: the last ones reduced the time involved in the search. The results suggest that the exchange of information in the proposed dACSs do not help the search, similar conclusion are reported in [2].
There are several issues which seem to be worth for further investigation. One issue deals with the effects of different uses of the information of solution arriving to promote a more direct use of the information in the algorithm. Another issue can be the investigation of search space characteristics and their relation to the algorithm performance.

Acknowledgments

This work has been partially funded by the Spanish Ministry of Science and Technology and the European FEDER under contract TIN2005-08818-C04-01 (the OPLINK project) and the proyect DIRICOM (P07-TIC-03044). We acknowledge the Universidad Nacional de La Pampa, Universidad Nacional de San Luis, and the ANPCYT in Argentina from which the first and second authors receive continuous support.

References

1. E. Alba (2005). Parallel Metaheuristics: A New Class of Algorithms. Wiley.
2. E. Alba, G. Leguizamon and G. Ordoñez (2007). Two models of parallel ACO for the minimum tardy task problem. Int. Journal High Performance Systems Architecture, 1:50.59.
3. E. Alba, J. Luna, L.M. Moreno, C. Pablos, J. Petit, A. Rojas, F. Xhafa, F. Almeida, M.J. Blesa, J. Cabeza, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, and C. León (2002). MALLBA: A Library of Skeletons for Combinatorial Optimisation, volume 2400 of LNCS, 927-932. Springer.
4. E. Alba and J. M. Troya (2001). Analyzing synchronous and asynchronous parallel distributed genetic algorithms. Future Generation Comput. Systems, 17:451465.
5. M. Boschetti and V. Maniezzo (2005). An ant system heuristic for the two-dimensional finite bin packing problem: preliminary results. Chapter 7 of book Multidisciplinary Methods for Analysis Optimization and Control of Complex Systems, 233-247.
6. M. Dorigo and L.M. Gambardella (1997). Ant colonies for the traveling salesman problem. BioSystems, 43(2):73-81.

7. M. Dorigo and L.M. Gambardella (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1(1):53-66.
8. L.M. Gambardella, E. Taillard, and M. Dorigo (1999). New Ideas in Optimization, chapter MACSVRPTW: A multiple ant colony system for vehicle routing problems with time windows, 63-76. McGraw-Hill.
9. E. Hopper and B. Turton (2001). A review of the application of meta-heuristic algorithms to 2D strip packing problems. Artificial Intelligence Review, 16:257-300.
10. F. Kruger, D. Merkle and M. Middendorf (1998). Studies on a parallel ant system for the BSP model.
11. J. Levine and F. Ducatelle (2004). Ant colony optimization and local search for bin packing and cutting stock problems. Journal of the Operational Research Society, (55):705-716.
12. R. Michels and M. Middendorf (1999). New Ideas in Optimization, chapter An ant system for the shortest common supersequence problem, pages 51.61. McGraw-Hill.
13. M. Middendorf, F. Reischle and H. Schmeck (2002). Multicolony ant system algorithms. Journal of Heuristics (Special issue on Parallel Metaheuristics, 8(3):305-320.
14. J. Puchinger and G. Raidl (2004). An evolutionary algorithm for column generation in integer programming: An effective approach for 2D bin packing. In X. Yao et al, editor, PPSN, volume 3242 of LNCS, 642-651. Springer.
15. M. Randall and A. Lewis (2002). A parallel implementation of ant colony optimization. Journal of Parallel and Distributed Computing, 62:1421-1432.
16. C. Salto, E. Alba and J. M. Molina (2008). Hybrid ant colony system to solve a 2-dimensional strip packing problem. International Conference on Hybrid Intelligent Systems, 708-713.
17. C. Salto, E. Alba and J. M. Molina (2009). Optimization Techniques for Solving Complex Problems, chapter Greedy Seeding and Problem-Specific Operators for GAs Solving Strip Packing Problems, 361-378. John Wiley & Sons, Inc.
18. C. Salto, J.M. Molina and E. Alba (2006). Evolutionary algorithms for the level strip packing problem. Proceedings of NICSO, 137-148.
19. P.Y. Wang and C.L. Valenzuela (2001). Data set generation for rectangular placement problems. EJOR, 134:378-391.

CHC and SA applied to The Distribution Of Wind Turbines on Irregular Fields

MARTÍN BILBAO[1] AND ENRIQUE ALBA[2]

[1] LabTEM, Universidad Nacional de la Patagonia Austral, Caleta Olivia, Argentina.
[2] Laboratorio de Ciencias de la Computación, Universidad de Málaga Málaga, España.
[1]{mbilbao}@uaco.unpa.edu.ar, [2]eat@lcc.uma.es.

Abstract. *In this article we analyze two kinds of metaheuristic algorithms applied to distribution of wind turbines in a wind farm. The basic idea is to utilize CHC (a sort of GA) and Simulated Annealing algorithms to obtain an acceptable configuration of wind turbines in the wind farm that maximizes the total output energy and minimize the number of wind turbines used. The energy produced depends of the farm geometry, wind conditions and the terrain where it is settled. In this work, the terrain is irregular and we will apply both algorithms to analyze the performance of the algorithms and the behavior of the computed wind farm designs.*

Keywords. *CHC, Simulated Annealing, Optimization, Wind Energy, Metaheurístics.*

1. Introduction

Wind energy is one of the most important alternative energies in the world. It is an economic, free, and clean energy and nowadays it can compete with other kinds of energy like fossil-fuel power production methods. The capital interest is to produce a maximum of energy at the same time as reducing the total cost of the wind farm. A farm is a set of wind turbines, every one being costly, whose position is a strategic decision to minimize the *wake effect* [1] in orden to maximize the produced energy. The goal in this paper is obtain a better configuration of the wind farm by using the conditions of the wind and the terrain given by the enviroment. In this work, we include a real wind distribution from Comodoro Rivadavia in Argentina taken in 2008 [2]. For that we need effective algorithms, that should be first evaluated before utilization.
Simulated Annealing [3] and Distributed Genetic Algorithms [4] have been used in the past to solve this kind of problem. In a previous work we used CHC and GPSO considered constant North wind [5] and CHC y Simulated Annealing considered the real wind distribution and flat terrain [6]. Now, we compare two scenarios using the real wind distribution and we consider irregular terrain. We analize the best farm configuration found, the fitness

value, the produced power, the efficiency, the performance of the algorithms in terms of their running, time and number of evaluations needed to obtain the best solution.
The rest of the article is structured as follows: Section 2 explains the wake model, the power model, and the cost model used. Section 3 will detail the real wind distribution, wind rose and field data. Section 4 describes CHC and SA the proposed algorithms. In Section 5 we will detail the objective function and the representation of wind turbine locations. In Section 6 we will detail the experimental studies and discuss on the results obtained; finally Section 7 summarizes the conclusions and future work.

2. Wind Farm Modelling

In this section we describe the mentioned inter-turbine wake effect model, the power model, and the cost model for our further mathematical manipulations. These are the basic components to deal with a realistic farm design, and they are combined together into an objective for the needed guidance of the function algorithms in their quest for an optimal farm configuration.

2.1 Wake Effect Model

The used model in this work is similar to the wake decay model developed by Katic [7]. Depending of the farm geometry, the wind turbine that is upwind of other wind turbine results in lower wind speeds than the one downwind, as shown in Fig. 1. The *velocity deficit* measures this effect [7]:

$$dV = U_0 - U_t = U_0 \frac{1-\sqrt{1-C_t}}{(\frac{1+2kX}{D})^2}, \tag{1}$$

where U_0 is the initial free stream velocity, U_t is the velocity in the wake at a distance X downstream of the upwind turbine, C_t is the thrust coefficient of the turbine, D is the diameter of the upwind turbine, and k is the wake decay constant. This model assumes that the kinetic energy deficit of interacting wakes is equal to the sum of the energy deficits of the individual wakes. Thus, the velocity deficit at the intersection of several wakes is:

$$U_t = U_0 \times \left[1 - \sqrt{\sum_{i=1}^{N}(1-\frac{U_i}{U_0})^2}\right], \tag{2}$$

where U_i is the free stream velocity of the individual wake, and N is the number of wind turbines in the wind farm.

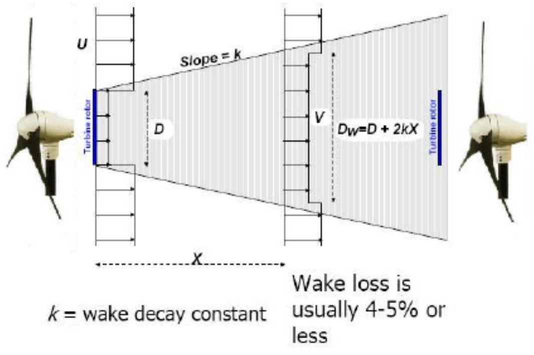


Fig. 1. Wake model for interaction between two wind turbines

2.2 Power Model

The previous wake model directly defines the power model, that is to be maximized. The power curve for the wind turbine under consideration is a Gamesa G47, whose power model (in KW) follows here:

$$P_i = \begin{cases} 0 & \text{for } U_x < 4m/s, \\ \rho \times A \times U_x^3 \times Cp & \text{for } 4m/s \leq U_x < 12.5m/s, \\ 700 \times Cp & \text{for } 12.5m/s \leq U_x \leq 25m/s, \\ 0 & \text{for } 25m/s < U_x \end{cases} \tag{3}$$

where U_x is the wind speed on the wind turbine, ρ is the density of the environment (1.23kg/m^3), A is the swept rotor area and C_p is the power coefficient of the wind turbine (0.45 in this case).

2.3 Cost Model

In our case, only the number of wind turbines influences the total cost to be minimized. The total cost per year for the entire wind farm, assuming a predefined and constant number of wind turbines, can be expressed as follows:

$$cost_{tot} = cost_{gy} \times N \times (2/3 + 1/3e^{-0.00174N^2}), \tag{4}$$

where $cost_{gy}$ represents the cost per wind turbine per year, and its value in this work is € 730,000. We consider three different cost, the cost of installation (€ 800 per Kw installed), € 80000 per cost of foundation and € 90,000 per cost of the tower.

3. Real Wind Distribution from Comodoro Rivadavia

In this section we introduce the real data obtained from Comodoro Rivadavia, Patagonia Argentina, and the process to obtain a good aproximation from the data for its later use in this work. Fig. 2(a) shows the frequency histogram and the relative frequency of the actual wind. We can see that the most probable frequency of the wind is between 2 and 5 m/s, and high probability has a range between 5 and 12 m/s. TheWeibull distribution is the most important probability distribution used in wind energy; it is usually used to approach the real wind data taken yearly (each 15 minutes) in our case. The process consists in obtaining a histogram of the wind with it frequency of ocurrence, relative and cummulative frequency. Then we apply a linear regression like least-squares to obtain a linear trend and calculate the parameters *k* and *b* of the Weibull distribution.

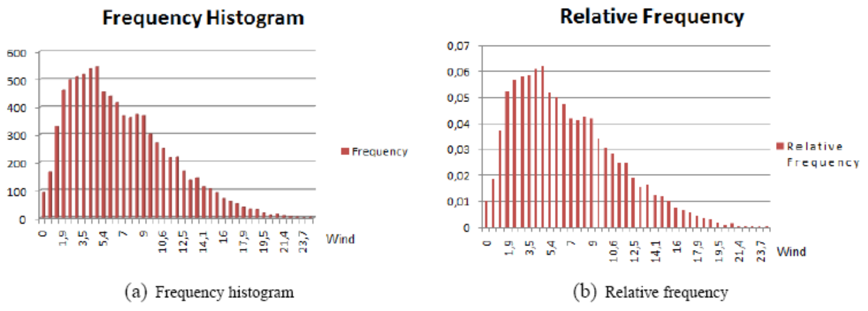


Fig. 2. Absolute Frecuency and Relative Frecuency of the real wind distribution

The probability of wind ocurrency is calculated as follows:

$$p(v) = (k/c)(v/c)^{k-1} \times e^{(v/c)^k}, \quad (5)$$

where *k* is a parameter form indicating if the wind speed tends to a particular value, and *c* is a parameter scale indicating how many winds are there in the environment. To obtain parameters *k* and *c* out of the natural measured data in the histograms we apply a linear regression whose form is $y = mx + b$, where m = k and $c = e^{-b/k}$

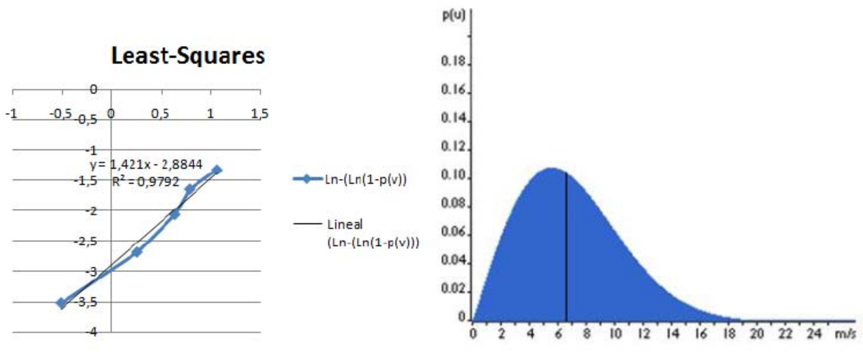
We obtain the linear trends for the independent variable x = ln(v) and the dependent variable y = ln−(ln(1− p(v))), being *v* the wind speed and *p(v)* the cummulative frequency of the wind *v*, as shown in Fig. 3(a).

We obtained, with the least-squares approach, the parameter k = 1.42, the parameter c = 7.53, and 98% correlative coefficient, and then we have completed the Weibull distribution needed for our algorithms and shown in Fig. 3(b). Table 1 shows a comparison between real data and the Weibull distribution to show their acurracy.

The total annual energy power obtained for each wind turbine can be calculated as follow:

$$P_{tot} = T \times \sum_{j=1}^{N} \int_{a}^{b} P_j(v) \times p_j(v) dx, \qquad (6)$$

where a y b are the cut-in and cut-out wind speed, T is the number of hours of the years (8760), p(v) is the weibull probability of the wind v and P_j is the power obtained for the wind turbine j.



(a) Least-Squares fitting for real data

(b) Weibull distribution

Fig. 3. Weibull Distribution

Table 1. Comparison with Weibull Aproach

Method	Mean Wind	Median
Real Data	6.79	6.1
Weibull Data	6.84	5.8

The resulting wind rose Fig 4 indicates the different frequency and direction of the wind. This rose is divided in to eight zones that indicate (in degrees) different cardinals point. In this scenario, the higher probability of ocurrence of wind direction is 270° (West direction). Thus our initial scenarios for evaluating the algorithms before a final real study will only consider in this work the wind coming from the West.

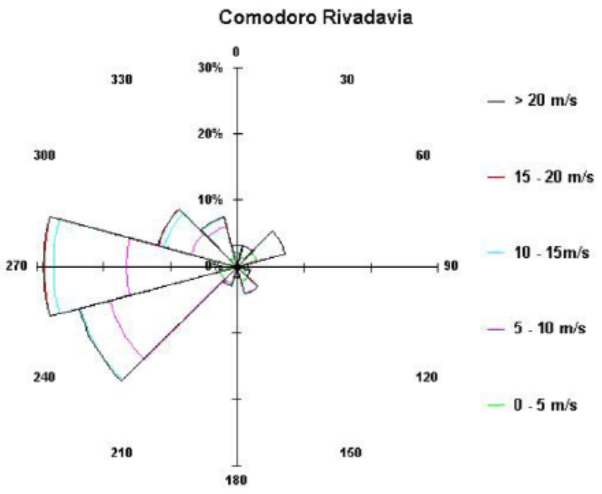


Fig. 4. Wind Rose of Comodoro Rivadavia (Patagonia Argentina)

4. Algorithms

In this section we will explain the algorithms that we will use to solve the optimization problem of optimally design a wind farm. We have selected two well-known algorithms, a good feature found in a previous work [3][5][6].

4.1 CHC

The CHC algorithm was designed to work with populations coded as binary strings. CHC is a type of genetic algorithm that does not use mutation to produce new solutions; insteads it uses a mechanism called *HUX* crossover. The selection of individuals to complete the next generation is under only an elitist approach between parents and children.The *R* best solutions are retained and will be present in the next generation. When stagnation in the population is detected, a cataclysmic method of restart is used. The population tends to be homogeneous due to the absence of mutation and the elitist approach because there is no diversity; in order to solve this problem

CHC implements a mechanism called *incest prevention*. The parents are selected randomly, but crossover takes place only if the individuals are not too close between them (Hamming distance) exceeds a certain threshold called *the threshold of incest*. As the population evolves, fewer individuals have the condition of not incest; in this case it is necessary to reduce the threshold. Every time that no change appears in the population (after one iteration) the threshold reduces in one unit.
The mechanism of crossover HUX also preserves diversity. This crossover copies in the two offspring all bits matched in both parents, and then copies half bits different in each offspring, such the Hamming distance between children and between children and parents is high. Once that the threshold of incest is 0, if *q* iterations pass without
any new solution has entered the population, it means that the population has converged and the algorithm has stagnated, thus requiring a restart. All individuals except the best are modified by a mutation by bit inversion with very high probability (in our case is 50%). Fig.5 shows an example of crossover HUX. It generates a mask with the common bits from the parents and non-common bits are assigned randomly to each child taking into account that each one must take half of the bits not common.
The pseudocode of the CHC algorithm is shown in Algorithm 1.

Algorithm 1 CHC

```
1: $t \leftarrow 0$; /* evaluation */
2: $initialize(Pa, Distance)$ /*Initialize the population and the distances */
3: while $not\ stop\ criterion(t, Pa)$ do
4:    $Parents \leftarrow selected(Pa)$; /* Selected parent */
5:    $Offspring \leftarrow HUX(Parents)$ /* Crossover HUX */
6:    $evaluate(Pa, Offspring)$ /*evaluate Offspring*/
7:    $Pa \leftarrow elitism(Offspring, Pa)$
8:    if $Pa$ $no\ change$ then
9:       $distance \leftarrow distance - 1$;
10:      if $distance == 0$ then
11:         $reset(Pa)$
12:         $initialize(distance)$
13:      end if
14:   end if
15:   $t \leftarrow t + 1$ /* One more generation */
16: end while
17: Return: best solution found.
```

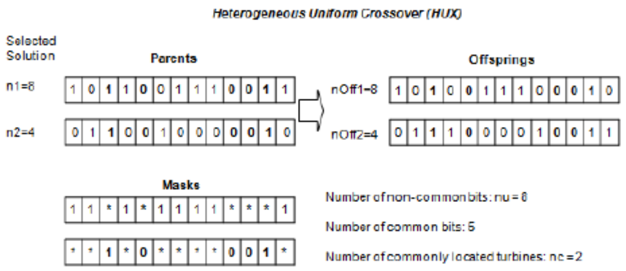

Heterogeneous Uniform Crossover (HUX)
Selected Solution
Parents
Offsprings
n1=8
n2=4
nOff1=8
nOff2=4
Masks
Number of non-common bits: nu = 8
Number of common bits: 5
Number of commonly located turbines: nc = 2

Fig. 5. Crossover HUX for CHC Algorithm

4.2 Simulated Annealing

Simulated Annealing (SA) is a metaheuristic for global optimization, aimed at locating a good approximation to the global solution. Simulated annealing is a generalization of a Monte Carlo method for examining the equations of states and frozen states of *n*-body systems[6]. In SA only one tentative solution exists. The initial tentative solution is created randomly. The perturbation of a solution to get a neighbor solution is done by choosing one position where the wind turbine exists and move it to other location. If the new solution is better than the old solution, it becomes the present tentative solution. If not, it can be used anyway but with a probability regulated by a decreasing temperature parameter called Boltzmann probability $e^{-((sn-sb)/T)}$, where sn is the present fitness value and sb is the old fitness value, T is the temperature parameter whose initial value is 100. After that, the temperature is decremented in each iteration, thus decreasing the posibility of a worse solution is accepted. The iterative process finishes when a stop criterion is reached (e.g.,maximum number of step), and returns the solution found.
The pseudocode implementing our simulated annealing solver is shown in algorithm 2:

Algorithm 2 Simulated Annealing

```
$s \leftarrow s0$; /* Initial state */
$sb \leftarrow s$; /* Initial best solution */
$k \leftarrow 0$; /* evaluation count */
$t \leftarrow 0$; /* Initial temperature */
while $k < kmax$ do
   $sn \leftarrow neighbor(s)$; /* Pick some neighbor */
   /* Is this a new best? maximizing */
   if $f(sn) \geq f(sb)$ then
      $sb \leftarrow sn$;
   end if
   if $Accept(sn, sb, t)$ then
      $sb \leftarrow sn$;
   end if
   $t \leftarrow UpdateT()$ /* Update temperature */
   $k \leftarrow k + 1$ /* One more evaluation done */
end while
Return: Best solution found.
```

5. Instantiating the Algorithms for the Problem

In this section, we will explain how our approach works: we will introduce the fitness function, the representation used, and the customizing of CHC and SA for the problem.

5.1 Objective Function

The objective function that we are maximizing is the annual profit got from the wind farm, defined as follows [8]:

$$profit = \left[st - \left(\frac{cost_{tot}}{P_{tot}}\right) \right] P_{tot} \pm G(x) \tag{7}$$

where *st* represents the estimated selling price for a KWh of electrical energy on the market in € (in this work it value is 0.1 €/KWh), P_{tot} represents the total expected energy output (kWh) of the wind farm per year, and $cost_{tot}$ is given by equation 4. The number of wind turbines is unknown and here also to be found by the used optimization algorithms.
The penalty function G(x) depends of the number of wind turbines included in the penalty zone, in this case we substract to fitness function the value calculated as follow:

$$G(x) = \left(\frac{k}{q}\right) * profit \tag{8}$$

where k is the number of the wind turbines included in the penalty zone and q is the total of places included in the restricted zone.

5.2 Representation of Wind Turbines Locations

As other existing approches for the problem of Wind Energy Optimization we discretize the terrain in a matrix. A wind farm is logically divided into many small square like cells. Each cell in the wind farm grid can have two possible states: it contains a turbine (represented by 1) or it does not contain a turbine (represented by 0). A 10×10 grid is used here as the ground platform to place the wind turbines, and shown in Fig. 6. A binary string with 100 bits represents the location of the wind turbines in the wind farm. There are 2^{100} candidate solutions. The width at each cell, in the center of which a turbine would be placed, is equal to five times rotor diameter, 5D (or 235 m). Thus, the resulting dimension is 50D × 50D. The 5D square grid size also satisfies the rule of thumb of spacing requirements in the vertical and horizontal directions.

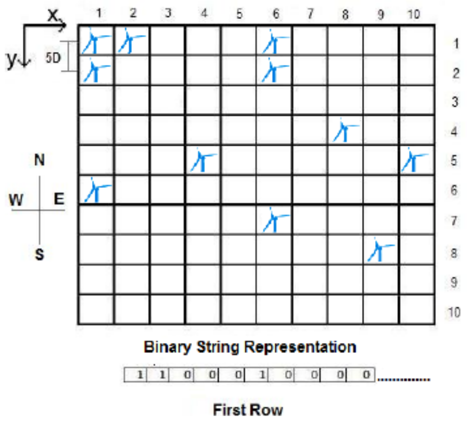


Fig. 6. Example of wind farm layout and the binary string representation

In this work we use an irregular terrain to evaluate the performance of the algorithms in this case, Fig. 7. shows the irregular terrain considered.

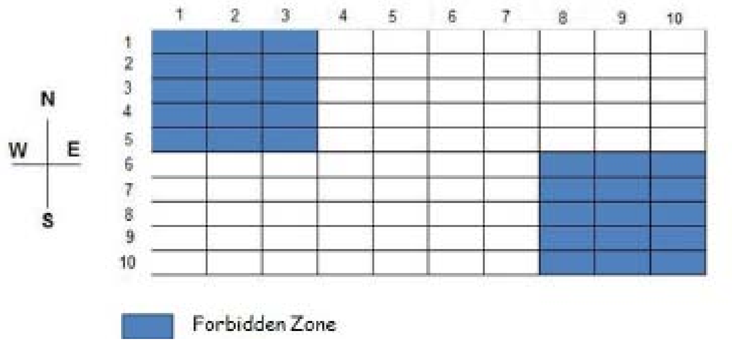


Fig. 7. Irregular terrain considered

5.3 Customizing algorithms for the problem

In this problem, SA was developed as follows: the individual consists of a binary vector $x_i = (x_{i1}, x_{i2},... ,x_{in})$ representing the terrain (10 × 10) where the wind farm will be installed; each element x_{ij} can have a wind turbine (represented by 1) or be empty (represented by 0). In this particular case (10 × 10) the individual has a length (n)
of 100 elements.
CHC was developed as follows: each individuals consists of a binary vector $x_i = (x_{i1}, x_{i2},... ,x_{in})$ in the same representation than SA, and the same criteria for the positioning of the wind turbines

6. Experimental Study

In this work we investigate two farm scenarios and we use the real wind distribution of Comodoro Rivadavia city. Our aim is to analyse two different kind of irregular terrain and try to generalize our conclusions to guide designer in similar configurations. Fig. shows the different kind of terrain used in this work.
We show the different configurations for each case with the average fitness values, standard deviation of the fitness, total annual power output, average power output, number of wind turbines, average efficiency of the park, average execution time of each algorithm and the number of evaluation needs to find the better solution. We have also computed a statistical study comparing the average fitness values, and execution time of each algorithm and we calcule the *p-value* with the *Kruskal-Wallis* test to conclude if it exists statistical significance between average fitness values and between average execution times. Each algorithm was executed 30 independent times with a stop criteria of 5,000,000 evaluations. All the algorithms are executed

in a MultiCore 2× QuadCore 2 GHz and for the implementation of the algorithms we have used the library of optimization MALLBA [9].
For each scenario we used the properties of wind turbines and the parameters of the each algorithm shown in Table 2.

Table 2. Property of wind turbines and parameters used in CHC and SA

(a) Wind Turbine Property

Description	Parameter	Value
Nominal Power	P	700 KWh
Rotor Diameter	D	47 m
Trust Coefficient	Ct	0.88
Wake Decay Constant	k	0.11
Cut-in Velocity	V_i	13 km/h
Cut-Out Velocity	V_p	90 km/h

(b) Parameters of CHC

Description	Value
Population Size	128
Crossover	HUX
Cataclismic Mutation	Bit Flip 50%
Preserved Population	5%
Initial Threshold	25% of instance size
Convergence Value Q	1
Selection of Parents	Randomly
Selection of New Generation	Elitist

(c) Parameters of SA

Description	Value
Temperature decay	0.99
Initial temperature	100
Probability of Mutation	0.3%

6.1 Scenario *(a)*: Northwest irregular terrain

For this scenario we have executed both algorithms (CHC and SA) with the parameters shown in Table 2(b) and 2(c) respectively, and we obtained the best configuration of the farm ilustrated in the Fig. 8 and the numerical values shown in Table 3.

Table 3. Results of scenario *(a)*

Description	CHC	SA
Average Fitness Values (€)	**1,9658e + 08**(± 8,2909 e+06)	1,9625 e+08 (± 8,3559 e+06)
Average Power Output (KWH)	**12,624.69**	12,406.89
Farm Coefficient (%)	**42.12**	41.93
Number of Wind Turbines (N)	42	42
Average Execution Time (s)	**26.95**	45.25
Average Evaluation of Best Solution Found	**2,335,749**	2,526,919

In this scenario CHC obtained better average fitness value, better power output and better efficiency. CHC needs less execution time and less evaluations to find the best solution than SA. We calcule the *p-value* with the *Kruskal-Wallis* test for the average fitness values and it value is 0.18e−04. This value is smaller than 0, 05, so we conclude that it exists statistical significance between average fitneses and that CHC is more accurate than SA. The *p-value* for the average execution time is 0.46e−07, it is smaller than 0.05, so we conclude that it exists statistical significance between average execution times and CHC is more faster than SA.
The configuration of the farm found for each algorithms is ilustrated in Fig. 8. We can see that the solution for CHC and SA uses 42 wind turbines and they are aligned in rows keeping a constants distance between them, and in an orthogonal position with respect to the wind direction.

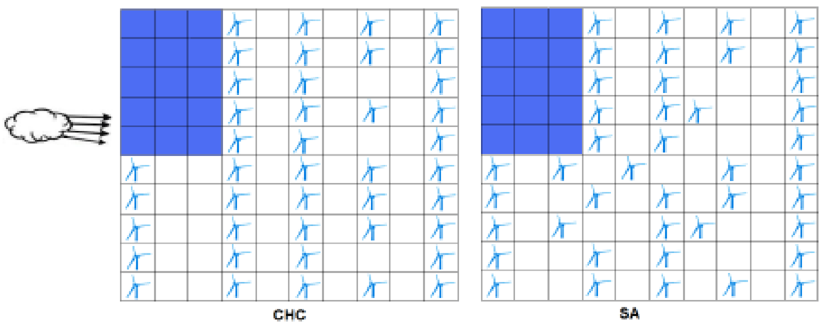


Fig. 8. Best configuration of the park for the two algorithms in scenario *(a)*

6.2 Scenario *(b)*: Northwest and Southwest irregular terrain

For this scenario we have executed both algorithms CHC and SA with the parameters shown in Table 2(b) and 2(c) respectively, and we obtained the best configuration of the wind farm ilustrated in the Fig. 9, with the numerical values shown in Table 4

Table 4. Results of scenario *(b)*

Description	CHC	SA
Average Fitness Values (€)	**1, 4458e + 08**(± 7,1458 e+06)	1,4225 e+08 (± 7,2296 e+06)
Average Power Output (KWH)	**10, 745.11**	10,444.2
Farm Coefficient (%)	**43.85**	41.44
Number of Wind Turbines (N)	**35**	35
Average Execution Time (s)	**61.15**	67.25
Average Evaluation of Best Solution Found	**4, 154, 111**	4,214,245

In this scenario CHC obtained the best average fitness value, better power output and better efficiency again. We calculed the *p-value* with the *Kruskal-Wallis* test for the average fitness values and it is smaller than 0.05, so we conclude that it exist stadistical significance between average fitnees values. The *p-value* for the average execution time is 0.002, it is smaller than 0.05, so we conclude that it exists statistical significance between average execution times.
The best configuration of the wind farm found for each algorithms is ilustrated in Fig. 9, where we can see that the number of wind turbines for CHC and SA is 35, they forming two rows in the center and in the opposite way with the wind sense.

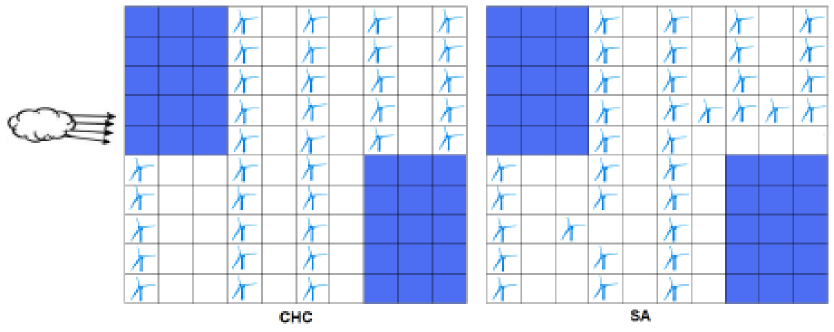


Fig. 9. Best configuration of the park for the two algorithms in scenario *(b)*

7. Conclusions and future works

We have here solved the problem of optimal placement of wind turbines in a wind farm with irregular terrain andthe objective to maximize the power energy produced with the less number of wind turbines to reduce the overall cost. CHC and SA algorithms are very competitive. In the first scenario CHC obtained better values in average fitness values, average efficiency and average power output than SA. Both obtained similar final configuration of the wind farm but SA did it in more execution time and more number of evaluations. In the second scenario CHC obtained better preformance in the majority of metrics. As a future work we will consider additional farm models, including more real world factors, such as terrain effect and the esthetic impact. Also, we intend to study the scalability of this problem with bigger instances of the wind farm and new parameters of the wind turbines. Finally we plan to solve this problem as multiobjective consider two contrast function, the cost of design the wind farm and the produced energy.

Acknowledgements

We here acknowledge partial funding from Project UNPA-29/B105, University of Patagonia Austral Argentina, DIRICOM Project N° P07-TIC-03044 and M* Project N° TIN2008-06491-C04-01, University of Malaga Spain.
M. Bilbao acknowledge the co-operation of the University of Malaga for providing new ideas and constructive criticisms. Also to the University of Patagonia Austral, and the ANPCYT (National Agency to Promote Science and Technology) from which we receive continuous support.

References

1. J.F. Manwell, J.G. McGowan and A.L. Rogers (2003). Wind Energy Explained-Theory, Design and Application, 1st ed., Reprint with correction, Jhon Wiley & Sons Ltd., 384, 44.
2. http://www.eeolica.com.ar/
3. M. Bilbao, E. Alba (2009). "Simulated Annealing for Optimization of Wind Farm Annual Profits" - 2nd International Symposium on Logistics and Industrial Informatics, Linz, Austria.
4. H.S. Huang (2007). "Distributed Genetic Algorithm for Optimization of Wind Farm Annual Profits". Intelligent Systems Applications to Power Systems. ISAP. International Conference on Volume, Issue, 5-8 Nov., 1-6.
5. M.Bilbao, E. Alba (2009). "GA and PSO Applied to Wind Energy Optimization". CACIC 09, Jujuy, Argentina.
6. M. Bilbao, E.Alba (2010). "CHC and SA Applied to Wind Energy Optimization Using Real Data". CEC 10, Barcelona.
7. I. Katic, J. Hojstrup and N. O. Jensen (1986). "A Simple Model for Cluster Efficiency", European Wind Energy Association Conference and Exhibition, Rome-Italy, 407-410, 7-9 October.
8. U. A. Ozturk and B. A. Norman (2004). "Heuristic methods for wind energy conversion system positioning", Electric Power Systems Research, vol.70, 179-185.
9. E. Alba, F. Almeida, M. Blesa, C. Cotta, M. Díaz, I. Dorta, J.Gabarró, C. León, G. Luque, J. Petit, C. Rodríguez, A. Rojas, F. Xhafa (2006). "Efficient Parallel LAN/WAN Algorithms for Optimization. The MALLBA Project", Parallel Computing 32(5-6):415-440.

X

Distributed and Parallel Processing Workshop

Predictive and Distributed Routing Balancing (PR-DRB)

CARLOS NÚÑEZ[1], DIEGO LUGONES[1], DANIEL FRANCO[2] AND EMILIO LUQUE[2]

Computer Architecture and Operating Systems Department, Universitat Autónoma de Barcelona, España.
[1]{carlos.nunez, diego.lugones}@caos.uab.es.
[2]{daniel.franco, emilio.luque}@uab.es.

Abstract. *Imbalance in traffic workload can produce network congestion. It also can raise latency values; decrease the throughput and thus the overall parallel system performance. Parallel applications show repetitive behavior during their execution. This repetitiveness allows their characterization and to obtain its representative communication patterns. This work presents the Predictive and Distributed Routing Balancing (PR-DRB), a new method developed to control network congestion by means of alternative paths, traffic and load distribution, in order to keep latency values low. PR-DRB monitors latency values at intermediate routers, chooses the best alternative paths for each situation and then saves this information. Also, conflictive communication pattern is identified and saved, in order to re apply the best solution when similar situation is detected again. Experimental results show that the predictive approach could be used to improve performance.*

Keywords. *Interconnection network, uniform latency, predictive routing, parallel applications, high performance computing, application aware routing.*

1. Introduction

The behavior of scientific applications that runs on a High Speed Interconnection Network (HSIN) could be described as a set of process implicitly assigned to each processor. The cost of the communications and their power consumption is higher than those of the processors, and it should be addressed by congestion control mechanism [1]. An inappropriate traffic load distribution can lead to situations where some portions of the network are congested while others remain idle. This is known as a hotspot. To alleviate this situation congestion control mechanisms have been developed, such as those that improve throughput by using resources properly [1]. Another example could be the adaptive routing algorithms which are used to

This research has been supported by the MEC-MICINN Spain under contract TIN2007-64974. Furthermore, we thank OPNET Technologies, Inc. for providing us the OPNET Modeler licenses

dynamically manage available resources in order to reduce congestion. This work presents the Predictive and Distributed Routing Balancing (PR-DRB), a new adaptive routing strategy. PR-DRB is based on alternative paths usage against congestion. This approach pretends to reduce latency values and to improve bandwidth under congestion, for repetitive traffic patterns. These improvements are achieved by using historical information about the conflictive communication pattern. The information saved will be used in future similar communications.
PR-DRB is based on the DRB [2] algorithm, but enhanced with the predictive monitoring and logging modules. The proposed model has three phases: monitoring, congestion and, conflictive pattern detection and congestion control. This work is also based on parallel applications repetitiveness [3]. During monitoring phase message latency information is saved. Besides that, the conflictive pattern that caused congestion is also saved. Once a message reaches its destination, an acknowledge message (ACK) is sent to the source node to inform these situations. Then, the source node is able to start path opening procedures based on latency values. If the notified situation was already analyzed, then the best saved solution is used and the historical database is properly updated.
When a congestion situation is detected DRB adapt itself by opening new alternative paths. This process is executed until a good global latency value is found and is a time consuming task. This work pretends to save the best solution encountered for a congestion situation, and re apply it when similar situations occur.
The rest of this paper is organized as follows. In section 2 related works are presented. Section 3 shows PR-DRB methodology in more details. Section 4 shows the performance evaluation. Conclusions and future work are explained in section 5.

2. Related Work

Congestion control is based on monitoring, detection and further control. In order to evaluate congestion point to point latency [4], buffer occupation level [5] or backpressure [1] are generally used. Message Throttling [6] is a corrective technique based on source notification, to reduce or stop new packets injection until the congestion disappears. This technique improves buffers occupations but latency is penalized because messages must wait in source nodes until congestion is controlled.
Other techniques are based on buffer occupation level at intermediate routers [5]. Buffer management implementation is also simple. However, good performance is not achieved because packet flows are locally reallocated to avoid contention but congestion sources are not controlled. There are techniques based on adaptive routing such as those in [2], [7], [8]. They use alternative paths in order to inject messages. Major adaptive routing advantage is that congested area is avoided and message injection is upheld. Thus, global system performance is improved because traffic load is fairly distributed over

the network. Some disadvantages of the adaptive routing mechanisms are the overhead resulting from information monitoring, the path changing and the need to guarantee both: deadlock freedom and in-order packet delivery. This leads to a trade off situation between the monitoring speed and the amount of information to analyze. An efficient algorithm should get the best performance under adverse situations and avoid penalizations.
Studies of parallel applications in HPC reveal they have repetitive behavior, based on computing and communications phases [3].
The routing performance in HSIN greatly depends on the communication patterns used and the relation between the mappings of application nodes to processors. This leads to proper resource usage in HPC where the total costs are prohibited [9]. Some routing techniques use application information to help routing decisions, in order to minimize latency, bandwidth, flows per links among others, but statically [10].
In order to improve overall system performance, a technique capable to combine adaptive routing and application communication patterns is needed. This combination will be helpful by re applying best saved solutions to already analyzed situations.

3. Predictive and Distributed Routing Balancing

PR-DRB seeks better response time than DRB by using historical communication information. PR-DRB uses metapath concept as alternative paths to the original to send messages. Metapath configuration defines how an alternative path is created and when it is to be used. PR-DRB phases are shown in **Fig. 1**. **Fig. 1 (a)** shows detection and logging of latency and the pattern causing congestion. **Fig. 1 (b)** shows metapath configuration and **Fig. 1** (c) shows alternative paths conforming the multi-step path (MSP). Monitoring phase logs a message latency value through its path to destination.
Congestion is detected at intermediate routers when latency value surpasses a threshold. Also, the conflictive pattern causing congestion and the source/destination involved are logged. The amount of alternative paths opened is governed by latency values logged during a message traversal towards the destination, in order to properly distribute all messages through these new alternative paths. A three-step path *(Multi-Step Path, MSP)* is then built by selecting two intermediate nodes, one IN1 neighbor from the source node, and the other IN2 neighbor from the destination node. Thus, PR-DRB builds the alternative paths around the original path. Finally, latency information is used to decide the number of alternative paths and to distribute messages among these paths. Afterwards, information about latency values and alternative paths used are saved into the historical database. With these procedure best paths for each src-dst pair, under a particular congestion situation, are saved and these solutions can be directly used at a later time.

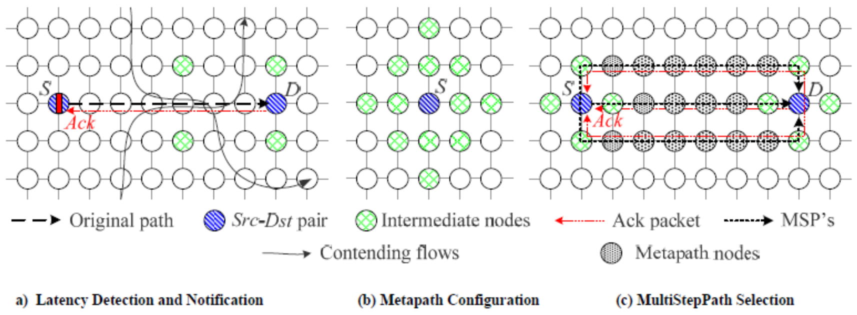


Fig. 1. PR-DRB algorithm phases

3.1 Monitoring and Congestion Notification Phase

Each intermediate router is in charge of monitoring the traffic passing through it, and is also responsible of logging the information that later would be sent to the source node via an ACK message. **Table 1** shows the Monitoring function.

Table 1. Monitoring and notification phase

```
Message Monitoring (Message M, Threshold, MSP)
/* At every intermediate PR-DRB router*/
Begin
For each message M at every hop,
    1.  If (First router in the path)
        –   Predictive = FALSE;
    2.  Accumulate latency (queue time)
        If (Latency > Threshold) AND NOT (Predictive)
            –   Identify contending flows in congestion
            –   Save contending flows (source/destination pairs) into the
                message
            –   Predictive = TRUE
    3.  Record message latency into the message
    4.  Forward message M. Continue to next router or final destination
    5.  At the destination node, latency and contending flows are sent back to the
        source into an ACK
End Monitoring and Notification
```

Contention latency is the time a message must wait at internal router buffers before it can continue towards the destination, because the buffer is blocked by other messages. This latency is then incremented at every router it traverses. Under contention the router saves information about the conflictive pattern at the buffers, to determine contending flows. Logging contending flows is done only at the first router, due to the fact that new alternative paths will control congestion for the analyzed flows. Once at the destination, an ACK message is sent to the source in order to inject new messages to the same destination using proper alternative paths. ACK messages have higher priority than other messages, and it can be considered negligible because it only transfers latency and conflictive communication pattern information. Logging and notification are carried out independently at each intermediate router with local information, including information about other messages currently in its internal buffers, which contributes to global network knowledge.

3.2 Metapath Configuration Phase

Dynamic metapath configuration is based on information collected during monitoring. The main goal of this phase is to determine the proper number of alternative paths to be used. These paths are created based on global latency value for each source/destination pair. Intermediate nodes (INs) are used for the alternative paths creation procedures. Congestion is controlled by increasing the available effective bandwidth between src-dst pairs. **Table 2** resumes metapath configuration. **Fig. 2** gives a general overview of PR-DRB working scheme. During application first phase, PR-DRB has high latency values (1) because it is searching alternative paths. At the end of phase 1 (2), latency is stable and the best solutions found are saved at the source node. Best solutions are identified when latency curve starts decreasing. Later phases do not reach its highest historical latency value. Here, PR-DRB has identified similar communication patterns again (3) and best paths saved are used (4).
PR-DRB approach is to maintain stable latency values during the whole application execution. **Table 2** shows the metapath configuration function.

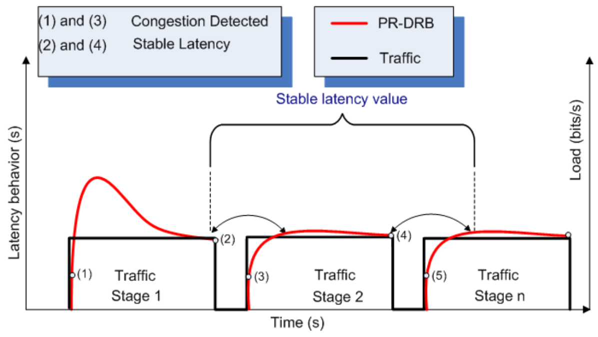


Fig. 2. PR-DRB process

Table 2. Metapath configuration function

```
Metapath Configuration.
/* Performed at source node every time and ACK message is received*/
Begin
    – Receive ACK with MSP latency and contending flows information
    – Calculate the metapath latency (MP*)

            $Latency\ (MP\ *) = \left( \sum Latency(MSPs)^{-1} \right)^{-1}$

    – If (Latency (MP *) > Threshold)
          If (Already exists a solution for the source/destination pair)
            – Search/retrieve the best MSP from the saved paths database
          Else
            – Increment INs number to provide new alternative paths
          End If
      Else If (Latency (MP *) < Threshold)
            – Decrease the number of INs to constrain metapath
            – Save metapath latency and source/destination nodes involved
      End If
End Metapath configuration.
```

3.3 Multistep Path Selection Phase

Each time a message is injected into the network, PR-DRB performs the multistep path (MSP) Selection. This phase is in charge of selecting a multistep path for each message. A distribution of communication load over the metapath is accomplished in order to perform the dynamic traffic balancing. Consequently, messages are distributed in the MSPs according to the latency in each case. Hence, paths having the lowest latency values are more frequently used, and they receive a greater number of messages. Path expansion is performed gradually. This stabilization process is costly in time. Given a source node with N alternative paths, let's be Lci (i:1..N) the latency recorded by path Ci. The alternative path Cx will be selected in the following injection according to the probability:

$$\rho(Cx) = \frac{\frac{1}{L_{Cx}}}{\left(\sum_{i=1}^{N} \frac{1}{L_{Ci}} \right)} \qquad (eq.1)$$

Find the best bandwidth/paths combination could take considerable time. Intermediate path expansion procedures could cause internal router contention, hence high latency values. PR-DRB keeps saving alternative paths opening information and the percentage of utilization of each path, for each source/destination used. All these information is saved because once similar situation is detected again; PR-DRB will try to avoid high latency during best paths finding procedures. **Table 3** shows the multistep path selection procedure.

Table 3. Multistep Path Selection

```
Multistep Path Selection
/*Performed at source node each time a message is injected*/
Begin
    –   Build accumulative function of distributions adding and normalizing
        Multistep Path (MSP) bandwidths.
    –   Generate a random number between [0,1)
    –   3. Select MSP using accumulative distribution function.
    –   4. Inject Message in the network.
            –   4.1. Build a multiple header with intermediate and final nodes.
            –   4.2. Concatenate header data.
            –   4.3. Inject message with PR-DRB format.
End  Multistep Path Selection
```

3.4 Integration of all phases

PR-DRB general working scheme is shown in **Fig.** When a source node injects a message into the interconnection network, a multistep path (MSP) is selected from the MSP table according to the respective path latencies. This message is injected without any other treatment because it is the first packet in a row and no other information or statistic about the path or network is available. The path with lowest latency is selected with higher probability. The PR-DRB multi-header message is forwarded to its destination through the intermediate routers. The delay suffered in the switch buffers (queuing latency), is recorded and stored in the message. If queuing latency values exceeds a threshold while still at intermediate routers, then contending flows are also recorded by the PR-DRB module. With this information, traffic pattern that caused congestion could be identified again in future communications. Once the message arrives to destination node, latency information as well as conflictive communication patterns found are sent back to the sender in an Ack.

When Ack reaches the source node, latency value is delivered to the metapath configuration module. This module configures the metapath with the alternative paths to be used according to the latency value. Also this module updates information about source-destination nodes and contending flows during high latency communication situations encountered, in order to have historical data about communications. With the information about communicating nodes, contending flows during a congestion situation and the solutions found to solve this congestion are saved. Metapath configuration module can predict future congestion based on previously similar situations analyzed.

Having predicted a congestion situation, metapath configuration module can speculate about which paths to open based on information already available. Because parallel application patterns show repetitive behavior in time,

metapath configuration can be reduced to find best solution during the first phase of execution, and then just identify similar phases and re-use best solutions. This behavior could save considerable time and communication efforts, such as path creation, ACKs notification, etc. PR-DRB node level operations have not a high overhead because: these operations are performed locally, they are simple (comparisons and accumulations for latency evaluation, saving small info about traffic), and they do not delay send/receive primitives.
As shown in **Fig. 3**, message is forwarded without any overhead when output port is free (thick arrows). Otherwise, packet is queued and latency is simultaneously accumulated. When output port is available and latency is lower than threshold, no more operations are needed and message is forwarded. The Ack generation is invoked only when congestion is detected, and its operations are performed when messages are waiting in the queue. Hence, computing these operations is performed concurrently with packet delivery, as shown in Monitoring in **Fig. 3**.
Deadlock freedom is ensured by having a separate escape channel for each phase. With two intermediate nodes, one escape channel is used (if required) from S to IN1, another from IN1 to IN2, and a third one from IN2 to D. This way, each phase defines a virtual network, and the packets change virtual network at each intermediate node.
Although each virtual network relies on a different escape channel, they all share the same adaptive channel(s). The use of adaptive routing algorithms can cause out of order delivery of packets. If an application requires in-order packet delivery, a possible solution is to reorder packets at the destination node using the well known sliding window protocol, as used in other routing policies like [7]. The following section presents the performance evaluation of PR-DRB policy.

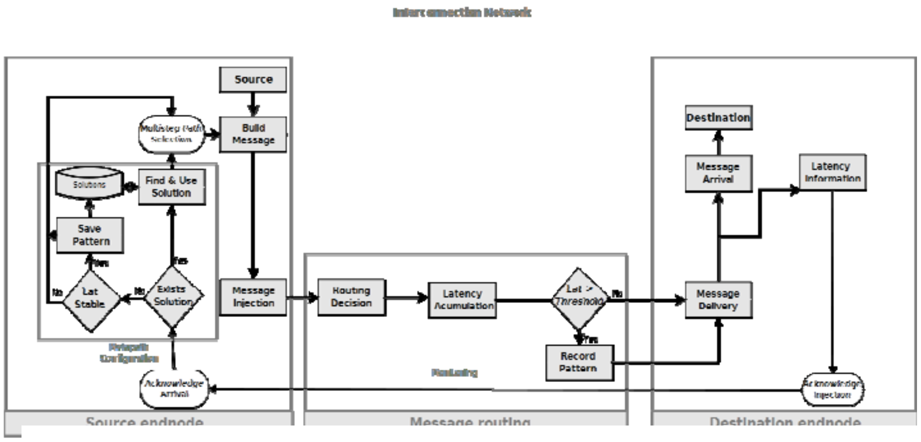


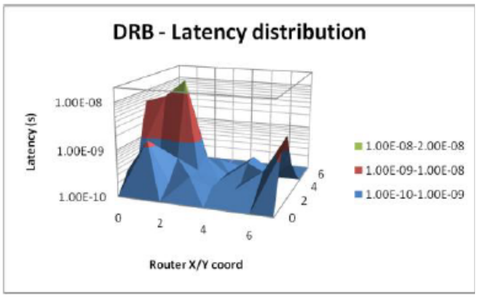
Fig. 3. PR-DRB Algorithm with all phases integrated

4. PR-DRB evaluation

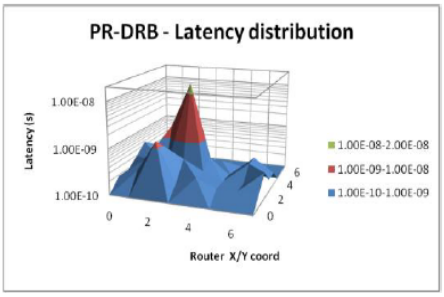
This section presents the performance evaluation of PR-DRB. Evaluation is designed to perform a network response analysis under hotspot traffic patterns in order to evaluate the PR-DRB dynamic behavior and the traffic load distribution over the network. We designed the Hotspot experiment to analyze and compare PR-DRB against the original DRB [2] algorithm. Here, we established some fixed destinations in order to increase the traffic in a particular network area and force saturated paths. In addition, the remainder network nodes inject uniform load in order to create a background or noise traffic over the 64 nodes network.
The PR-DRB operations and modules, together with network components (i.e. switches, links and end nodes) were modeled using the standard simulation and modeling tool OPNET Modeler [11]. OPNET provides a Discrete Event Simulator (DES) engine and offers a hierarchical modeling environment with an enhanced C++ language. This suitable environment allows defining network components behavior by a Finite State Machine approach (FSM), and it supports detailed specification of protocols, resources, applications, algorithms, and queuing policies. The simulations were conducted for a 64 nodes network arranged in an 8x8 mesh topology. We have assumed virtual Cut-through flow control and several standard packet sizes. Link Bandwidth was set to 2Gbps, packet size was set to 1024 bits and the size of routers buffers was 2MB.

4.1 Hotspot Analysis

Fig. 4 (a) and **(b)** show latency map of the mesh network under evaluation. Latency surface represents the average contention latency at buffers. **Fig. 4 (a)** shows the behavior of the original DRB algorithm, where high values of latency can be seen under congested areas. Also, load distribution at routers (x,y) (0,1), (6,2) and (6,4) are considerable high, because DRB uses these routers in its alternative paths. **Fig. 4 (b)** shows the latency map for PR-DRB; where its highest value is lower than the original DRB. Better load distribution is accomplished by PR-DRB compared to DRB, because PR-DRB has directly applied best solutions already saved and unnecessary load at routers are avoided. **Fig. 5 (a)** shows latency during application initialization to analyze algorithms reactions under initial traffic injection**.** On average, PR-DRB performs better than DRB. This is because PR-DRB reaches better global latency values faster and without penalizing throughput. **Fig. 5 (b)** shows average latency values for the entire mesh network during initialization phase of communications, in order to analyze algorithms response under initial traffic injection. On average, PR-DRB outperforms DRB because it reaches better global latency values in less time. Throughput is not penalized whatsoever with latency gains of PR-DRB. **Fig. 5 (b)** also shows that DRB has an initial latency raise due the fact that it is opening alternative paths in order to control a particular congestion situation. Recall that PR-DRB will behave similar to DRB under first phase of parallel applications execution, because it will be learning from the alternative opening paths procedures. In next phases of application repetitive behavior, PR-DRB will apply directly best solutions encountered previously. Between time 0 and 0.5 of **Fig. 5 (b)** we can see latency gains obtained by PR-DRB. From time 0.5 latency values of both algorithms tend to become stable and converge.

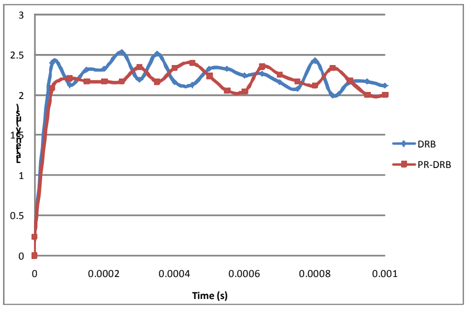


(a)

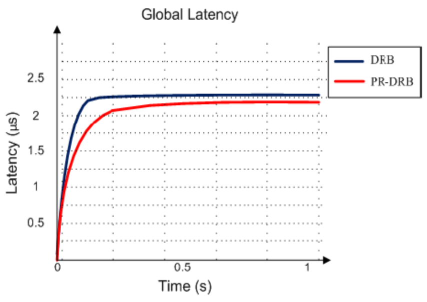


(b)

Fig. 4. Latency map



(a)



(b)

Fig. 4. Entire network latencies, instantaneous (a) and average (b)

5. Conclusions

In this paper we proposed the Predictive and Distributed Routing Balancing PR-DRB. PR-DRB limits performance degradation produced by packet contention in network resources. This strategy uses alternative paths under congestion situation in order to reduce latency values and increase bandwidth availability, by considering time as well as traffic dynamic behavior constraints. Parallel applications running on an HSIN possess repetitive behavior, and PR-DRB is capable of learning about repetitive traffic patterns and save information for later use. PR-DRB has been developed to fulfill the design objectives for cluster interconnection networks. These objectives are all-to-all connection, and low and uniform latency between any pair of nodes under any message traffic load. The proposed method is also in line with current approaches used in commercial interconnects (as InfiniBand).
Our policy allows heavier communication load in the network, or in the case cost-bounded clusters, PR-DRB allows using less network components, because those resources are more efficiently handled. The evaluation performed to validate PR-DRB has revealed very good improvements in latency. Saturation is reduced allowing the use of the network at higher loads. We have shown that PR-DRB is a fast and robust method with a very low overhead. Additionally, PR-DRB is useful for permutation and bursty communication patterns, which are commonly created by parallel applications and can produce the worst hot-spot situations.
As a continuation of this work, we plan to predict future congestion situation based on latency trend, before congestion has effectively emerged into the network. Also, more experiments with different scenarios is to be done, such as more topologies as well as real representative scientific applications traces as input, in order to extend the proposal to a fully application aware technique.

References

1. Elvira Baydal, Pedro Lopez and Jose Duato (2005). "A Family of Mechanisms for Congestion Control in Wormhole Networks", IEEE Trans. Parallel Distrib. Syst., vol. 16, no. 9, 772-784.
2. D. Franco, I. Garcé and E. Luque (1999). "A new method to make communication latency uniform: distributed routing balancing", 210-219.
3. Wong, D. Rexachs and E. Luque (2009). "Parallel application signature," Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on, vol. 1, 1-4.
4. D. Lugones, D. Franco and E. Luque (2009). "Dynamic and Distributed Multipath Routing Policy for High-Speed Cluster Networks", 396-403.
5. P.J. Garcia et al. (2006). "RECN-DD: A Memory-Efficient Congestion Management Technique for Advanced Switching", Parallel Processing, International Conference on, vol. 0, 23-32.
6. Shihang Yan, Geyong Min and Irfan Awan (2006). "An Enhanced Congestion Control Mechanism in InfiniBand Networks for High Performance Computing Systems", Advanced Information Networking and Applications, International Conference on, vol. 1, 845-850.
7. Arjun Singh, William J. Dally, Brian Towles and Amit K. Gupta (2004). "Globally Adaptive Load-Balanced Routing on Tori", IEEE Comput. Archit. Lett., vol. 3, no. 1, 2.
8. Christopher J. Glass and Lionel M. Ni (1992), "The turn model for adaptive routing", SIGARCH Comput. Archit. News, vol. 20, no. 2, 278-287.
9. German Rodríguez, Ramon Beivide, Cyriel Minkenberg, Jesus Labarta and Mateo Valero (2009). "Exploring pattern-aware routing in generalized fat tree networks", 276-285.
10. Michel A. Kinsy et al. (2009). "Application-aware deadlock-free oblivious routing", 208-219.
11. OPNET Technologies, Opnet Modeler Accelerating Network R&D, 2008.
12. Xingfu Wu and Xian-He Sun (2000). "Performance Modeling for Interconnection Networks", High-Performance Computing in the Asia-Pacific Region, International Conference on, vol. 1, 380.
13. Timothy Sherwood, Erez Perelman and Brad Calder (2001). "Basic Block Distribution Analysis to Find Periodic Behavior and Simulation Points in Applications", 3-14.

Comparison of Communication/Synchronization Models in Parallel Programming on Multi-Core Cluster

ENZO RUCCI, ARMANDO E. DE GIUSTI, FRANCO CHICHIZOLA, R. MARCELO NAIOUF AND LAURA C. DE GIUSTI

Instituto de Investigación en Informática LIDI (III-LIDI) – School of Computer Science – Universidad Nacional de La Plata, Argentina.
{erucci, degiusti, francoch, mnaiouf, ldgiusti}@lidi.info.unlp.edu.ar.

Abstract. *Taking into account the increase in use of the multi-core cluster architecture, in this paper we analyze the use of the various communication models (message passing, shared memory, their combination) to efficiently exploit the power of the architecture.*
Smith-Waterman algorithm, whose parallelization is based on a pipeline scheme due to problem data dependence, is used as test case to determine the similarity degree of two DNA sequences.
Finally, future research lines are mentioned, aimed at optimizing the use of memory levels in the architecture.

Keywords. *multi-core cluster, hybrid communication model, pipeline, Smith-Waterman.*

1. Introduction

The study of distributed and parallel systems is one of the most active research lines in Computer Science nowadays [1][2]. In particular, the use of multi-processor architectures configured as clusters, multi-clusters, and grids, supported by networks with different characteristics and topologies has become generalized, for the development of both parallel algorithms and distributed Web services [3][4][5]. Cloud computing developments follow the same line [6].
The technological change, mainly based on multi-core processors, has created the need for researching mixed or hybrid models where shared memory and message schemes are in coexistence [7][8].
In this context, it is important to study the modeling of the behavior of this type of parallel systems, as well as develop new paradigms and tools for efficient application programming [9][10][11].

1.1 Multi-core cluster

A multi-core processor is formed by the integration of two or more computational cores within the same chip [12]. The reasons for their development are based on the energy consumption and heat generation problems that appear when the speed of a processor is escalated.
A multi-core processor increases the yield of an application by dividing the computation work among the available cores [13].
A cluster is a parallel processing system formed by a set of computers interconnected over some kind of network and that cooperate as if they were an "only and integrated" resource, regardless of the physical distribution of its components. Each "processor" may have different hardware and operating system, and it can even be a "multiprocessor" [14].

1.2 Study application

One of the areas of greatest interest and growth in the last few years within the field of parallel processing is that of the treatment of large volumes of data such as DNA sequences. The extensive comparison processing required for the analysis of genetic patterns demands a significant effort in the development of efficient parallel algorithms [15].
The center for all bioinformatic operations and analyses is partly held by Sequence Alignment, both for pattern searching among amino acid and nucleotide sequences, and for the search of phylogenetic relationships among organisms. The Smith-Waterman algorithm for local alignment is one of these methods; it focuses on similar regions only in part of the sequences, which means that the purpose of the algorithm is finding small, locally similar regions. This method has been used as the basis for many subsequent algorithms and is oftentimes used as basic pattern to compare different alignment techniques. If the length of the sequences involved are *N* and *M*, the complexity of the algorithm is *O(NxM)*. Thus, the problem is escalated as the square of sequence size [16].
Taking into account that sequences can have up to 10^9 nucleotides each, the time and memory required to solve this problem in a sequential manner is impracticable. This leads to the parallelization of the algorithm over powerful parallel architectures.

1.3 DNA Sequence Comparison on a Multi-core Cluster

Taking into account the increase in use of the multi-core cluster architecture, it is important to study new parallel algorithm programming techniques that efficiently exploit the power of the architecture by combining shared memory and message passing.
In particular, the approach of the application to study is attractive due to its complexity and the possibility of breaking down parallel algorithm

concurrency into “blocks” of different dimensions, which allows an optimal adaptation of the application to the multi-core cluster support architecture.
The architecture used in this paper is a Blade with 8 blades. Each blade has 2 quad core Intel Xeon e5405 2.0 GHz processors; 2 Gb of RAM memory (shared between both processors); and 2 X 6Mb L2 cache for each pair of cores [17][18]. This architecture allows a comprehensive analysis of the three approaches (messages, shared memory, and hybrid).
In Section 2, the Smith-Waterman algorithm is explained, together with the sequential and the parallel solutions used in this paper. In Section 3, the experimental work carried out is described, whereas in Section 4, the results obtained are presented and analyzed. Section 5 presents the conclusions and future lines of work in relation to this paper.

2. Smith-Waterman Algorithm Definition

This method allows aligning two DNA sequences by inserting *gaps* (if necessary) that are used to detect locally similar regions that may indicate the presence of a relation between both sequences, which is done by assigning a similarity score. If gaps are inserted, that is, certain elements of the sequences are not aligned to achieve a better overall alignment, a penalization is applied.
The algorithm calculates a similarity score between two sequences and then, if necessary, employs a backwards alignment process for an optimal result [14].
The following paragraphs explain the operation of the algorithm to find a similarity score between two DNA sequences.
Given two sequences: $A = a_1a_2a_3\ldots a_M$ and $B = b_1b_2b_3\ldots b_N$, a matrix H of $(N+1)$x$(M+1)$ is built, in such a way that the nucleotide bases that form sequence A label the rows (starting with 1), and those from sequence B label the columns (starting with 1). The following steps are applied to calculate the values of H that will yield the similarity score between A and B:

a. Start row 0 and column 0 of H with 0, as indicated in Equation 1.

$$H_{i0} = H_{0j} = 0 \qquad \text{for } 0 \le i \le N \text{ and } 0 \le j \le M \tag{1}$$

b. Calculate the value of H_{ij} $\forall i \in [1,.., N]$ and $\forall j \in [1,..,M]$ by means of Equation 2. This value indicates the maximum similarity between two segments ending in a_i and b_j, respectively.

$$H_{ij} = \max\begin{cases} 0 \\ H_{i-1,j-1} + V(a_i, b_j) \\ C_{ij} \\ F_{ij} \end{cases} \tag{2}$$

- $V(a_i, b_j)$ is the matching function that indicates the score obtained for matching a_i with b_j. It is based on a table of values called *substitution matrix* that describes the probability of a nucleotide base from sequence *A* at position *i* to occur in sequence *B* at position *j*. The most common matrix is the one that rewards with a positive value when a_i and b_j are identical, and punishes with a negative value otherwise.

- C_{ij} is the score in column *j* considering a gap, and is calculated with Equation 3.

$$C_{i\,j} = \max_{1 \le k \le i}\{H_{i-k,j} - g(k)\} \tag{3}$$

- R_{ij} is the score in row *i* considering a gap, and is calculated with Equation 4.

$$R_{i\,j} = \max_{1 \le l \le j}\{H_{i,j-l} - g(l)\} \tag{4}$$

- *g(x)* is the penalization function for a gap of length *x*, and is obtained with Equation 5, *q* being the penalization applied for opening a gap and *e* the penalization for prolonging it.

$$g(x) = q + r\,x \qquad (q \ge 0; e \ge 0) \tag{5}$$

c. The similarity score is obtained as shown in Equation 6.

$$G = \max_{(0\le i\le N)(0\le j\le M)}\{H_{i\,j}\} \tag{6}$$

d. Based on the position in matrix *H* where the value *G* was found (representing the end of the highest-scoring alignment between both sequences), a backwards process is performed to obtain the pair of segments with maximum similarity, until a position whose value is 0 is reached, this being the starting point of the segment.

2.1 Sequential Solution of Smith-Waterman Algorithm

In this section, the sequential solution of Smith-Waterman algorithm is analyzed with the purpose of determining the similarity score between two DNA sequences. This means that the backwards process is not taken into account when obtaining the segment that represents the optimal alignment (step *d* of the algorithm explained in the previous section is not performed).

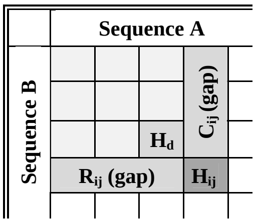


Fig. 1. Data dependency scheme

Figure 1 shows the data dependency that exists for calculating matrix values. To obtain $H_{i,j}$, the result of $H_{i-1,j-1}$ (H_d in Figure 1) is required, and the score must be known when considering a gap in row i and another one in column j. This restriction allows calculating H values from top to bottom and left to right ($H_{11}, H_{12}, H_{13}, \ldots H_{21}, H_{22}, H_{23}, \ldots..$).
Taking into account that step d of the algorithm is not carried out, matrix H does not have to be stored in full, all that is needed is:

- A vector h of length $M+1$ that at each position keeps the value obtained in the last processed row over that column. Equation 7 shows the values for h corresponding to the example shown in Figure 1.

$$h_k = \begin{cases} H_{i,k} & k < j-1 \\ H_{i-1,k} & k \geq j-1 \end{cases} \tag{7}$$

- An element e to temporarily store the last value calculated in the row that is being processed. In Figure 1, $e = H_{i,j-1}$.

- A vector c of length $M+1$ that at each position keeps the maximum score considering a gap in that column. Equation 8 shows the values for c corresponding to the example shown in Figure 1.

$$c_k = \begin{cases} C_{ik} & k < j \\ C_{i-1,k} & k \geq j \end{cases} \tag{8}$$

- An element r that keeps the maximum score considering a gap in the row that is being processed. In the example shown in Figure 1, $r = R_{i,j-1}$.

2.2 General Parallel Solution of Smith-Waterman Algorithm

Due to the dependency of data mentioned in the previous section, the problem needs to be solved by following a pipeline scheme, where stages S perform the same work over various consecutive nucleotide subsets of the first sequence (A in Figure 1). In each cycle, stage s_i (for $i \in [1, S\text{-}1]$) receives a data block from s_{i-1}, solves part of its work, and then sends these results to s_{i+1} (except for the last stage which does not need to send its results to any other stage). The first stage (s_0) only performs its work by sending partial results (corresponding to a block) to its successor.
An important aspect of this solution is selecting the number of elements (BS) from sequence B that form the data blocks that are sent from one process to another, taking into account that:

- Pipeline parallelism is exploited to its maximum capacity only after S-1 cycles have been processed. That is, when all stages have received work to do. The larger the BS, the longer the time required to fill the pipe, and therefore, the lower its exploitation. From this point of view, BS should tend to 1.

- If the size of *BS* is very small, the stages spend more time communicating partial results than actually processing information. From this point of view, *BS* should tend to *N*.

A suitable block size should be found, so that data communication and data processing can be done simultaneously. The optimal size does not only depend on the architecture used, but also on the communication model implemented.

2.2.1 Message Passing as Communication Model

In this case, each pipeline stage is carried out by a different process p_i (for $i \in [0, S\text{-}1]$), and partial results are communicated by sending messages between consecutive processes. The first sequence (*A* in Figure 1) is distributed by p_0 among the *S* processes that form the pipeline.

2.2.2 Shared Memory as Communication Model

In this case, each pipeline stage is carried out by a different thread t_i (for $i \in [0, S\text{-}1]$). Instead of communicating partial results through message passing, these are kept in the shared memory as a single structure (as in the sequential algorithm). Consecutive threads are synchronized to indicate that work with a new data block can begin.

2.3 Hybrid Parallelization of the Algorithm by Integrating Message Passing and Shared Memory

When using a hybrid architecture, the different memory levels (among cores in a same blade) and the interconnecting network (among cores in different blades) should be considered to determine the optimal size *BS*. This leads to a solution that combines the use of message passing with shared memory.
This hybrid solution is based on the use of a pipeline of *P* stages as the one described in Section 2.2.1, each of these stages using a pipeline of *T* phases as the one detailed in Section 2.2.2.
When each process p_i begins (for $i \in [0, P\text{-}1]$), it generates *T-1* threads to jointly solve the data blocks corresponding to the different cycles. Thus, there are *P*x*T* threads (all *P* processes plus all *T-1* threads generated by each of them), which means that the set of nucleotides from the first sequence (*A* in Figure 1) is equally distributed among *P*x*T* threads.
When process p_i (for $i \in [0, P\text{-}1]$) needs to solve a data block (with BS_{mp} elements), it divides it in sub-blocks of BS_{sm} nucleotides each to be solved by the pipeline corresponding to that process. To take advantage of the features of the architecture, the optimal BS_{mp} and BS_{sm} values have to be determined for each case.

3. Experimental Work

In this paper, language C is used with OpenMPI and/or Pthreads libraries to handle message passing and threads, respectively.
As mentioned in the Section 1, a Blade with 8 blades, each with two 2.0 GHz quad core Intel Xeon e5405 processors, was used. Each blade has 2 Gb RAM memory (shared between both processors) and 2 x 6Mb L2 cache for each pair of cores.
Two types of tests were carried out:

- Using one blade of the Blade. Testing in this case purely parallel algorithms to determine suitable data block sizes.
- Using the entire architecture. Testing in this case the hybrid algorithm and the one that uses only message passing to compare both behaviors.

3.1 Tests with a single blade of the Blade

Tests were carried out on a single blade of the Blade (using the 8 cores) to analyze the behavior of the purely parallel algorithms described in Sections 2.2.1 and 2.2.2.
The tests carried out vary in sequence length (*N = 65536, 131072, 262144, 524288, 1048576*) and block size (*BS = 8, 16, 32, 64, 128, 256, 512, 1024, 2048*).
As a result of these tests, it was observed that the efficiency achieved by the algorithm that uses shared memory is slightly higher than the efficiency of the solution that uses message passing. This leads to the proposal of the hybrid algorithm described in Section 2.3 to fully exploit the architecture.
Also, the results obtained allowed determining the ideal values for BS_{sm} = 16 and BS_{mp} =128. Detailed results can be found in [19].

3.2 Tests with the entire architecture

In order to analyze the behavior of the hybrid algorithm, it is compared with the algorithm that uses message passing only, using the 8 cores with different numbers of blades (4 or 8). The same as in Section 3.1, sequence length varies (*N = 65536, 131072, 262144, 524288*). In the following paragraphs, the tests carried out are described.

- <u>*MP*</u>*:* the algorithm that uses only message passing is used with various block sizes (*BS = 128, 256, 512, 1024, 2048*).
- <u>*HY*</u>: the hybrid algorithm is used with a process p_i and 3 threads for each processor in each blade. That is, each shared memory pipeline uses an entire quad core processor. The value of BS_{sm} remains unchanged during the tests, and the size of the blocks in the message passing pipeline varies (BS_{mp} = 128*, 256, 512, 1024, 2048*).

4. Results

To assess the behavior of the algorithms developed when escalating the problem and/or the architecture, *efficiency* is analyzed (in this case, on homogeneous architectures, since all cores are equal) [1][2][20]. Equation 9 indicates how to calculate this metric, where *p* is the total number of cores used.

$$Efficiency = \frac{Speedup}{p} \qquad (9)$$

Figure 2 shows the efficiency achieved by the algorithms *MP* and *HY* detailed in Section 3.2 for the most significant block sizes (BS_{mp} = 128, 512 and 2048). For readability, only the results obtained when using all 8 blades of the architecture are shown, since when using only 4, a similar behavior is observed, with a slight increase in efficiency.

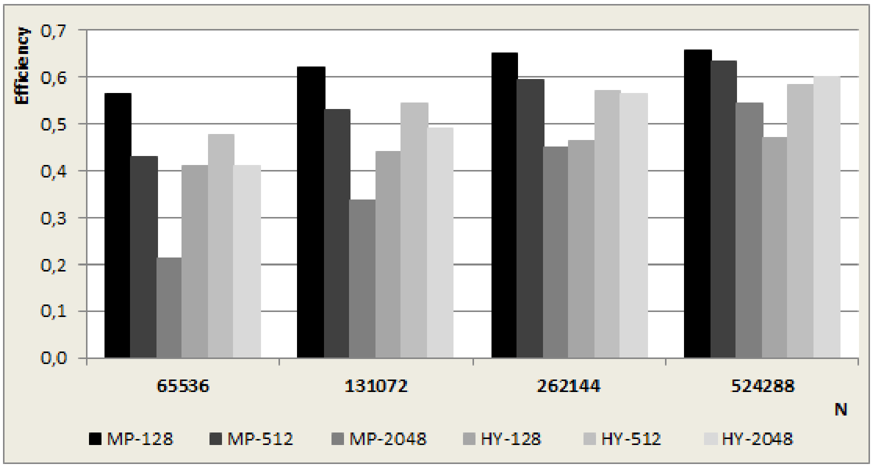


Fig. 2. Efficiency achieved by the algorithms *MP* and *HY* for different BS_{mp} and *N* values, using 64 cores (8 cores in 8 blades)

This chart shows that both algorithms increase their efficiency when the size of the problem increases (sequence length). On the other hand, these results confirm that, for algorithm *MP*, the ideal block size is 128. The hybrid algorithm (*HY*), however, tends to improve its efficiency when the size of the blocks increases, so that when comparing both algorithms using a block size of 2048, *HY* achieves a better efficiency.
Figure 3 presents a summary of the best efficiency achieved by each of the algorithms (*MP* and *HY*) when using 4 and 8 blades of the architecture. In the case of algorithm *MP (MP-4* and *MP-8* for 4 and 8 blades, respectively), it is achieved with a block size *BS* = 128. For *HY*, it is achieved when using BS_{mp} = 2048.

This chart shows that algorithm *MP* achieves a greater efficiency than the hybrid algorithm, and that the difference decreases as the size of the problem increases. On the other hand, as it is to be expected in most parallel systems, efficiency decreases when the number of cores used increases.
This figure also shows that when the total number of cores used increases, so does the difference between the efficiency achieved by *MP* and *HY*. Inversely, when the size of the problem increases, the gap decreases.

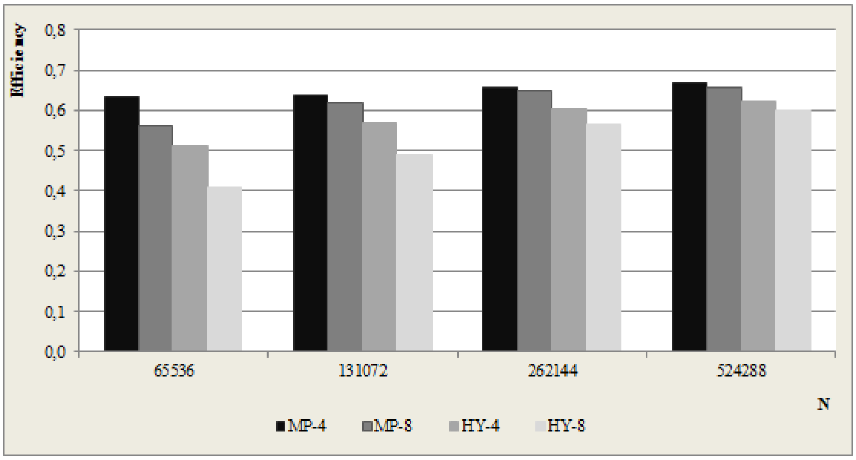


Fig. 3. Summary of the best efficiency achieved by each of the algorithms with 4 and 8 blades of the architecture

5. Conclusions and Future Works

In this paper, the Smith-Waterman algorithm is parallelized for the alignment of DNA sequences by means of a pipeline scheme due to the dependency of data that is inherent to the problem. The architecture used for the experiments is a multi-core cluster (8 blades with 8 cores each).
Given the characteristics of the architecture, the pipeline was initially implemented with two different communication models: message passing (*MP*) and shared memory (*SM*). The efficiency of both algorithms was compared when using a single blade of the architecture and a slight advantage of *SM* in relation to *MP* was observed.
Since the *SM* algorithm could not be used in the entire architecture (because there was no memory shared among the various blades), a third option was implemented (*HY*) using a hybrid communication model that combines message passing and shared memory. This version has a pipeline scheme among processes that communicate thorough message passing, and within each stage there is a shared memory pipeline to solve each data block.

The behavior of this algorithm was compared with that of *MP* using 4 and 8 full blades of the architecture. As a result, it was observed that *MP* achieves a greater efficiency than *HY*. This is because the optimal block size for *MP* (*BS* = 128) cannot be used in *HY* ($BS_{mp} >> 128$) pipeline because it would not be possible to generate enough sub-blocks to run the internal shared memory pipeline efficiently with its optimal block size ($BS_{sm} = 16$).
A future line of R&D is the analysis and optimization of hybrid solutions for certain types of problems, especially for those that support a composite parallel solution (combining more than one paradigm). On the other hand, the scalability of the problem discussed, while ensuring a certain efficiency level, is also of interest.

References

1. Grama, A., Gupta, A., Karypis, G., Kumar, V. (2003). “An Introduction to Parallel Computing. Design and Analysis of Algorithms. 2nd Edition”. Pearson Addison Wesley.
2. Jordan, H, Alaghband, G. (2002). “Fundamentals of parallel computing”. Prentice Hall.
3. Dongarra, J., Foster, I., Fox, G., Gropp, W., Kennedy, K., Torczon, L., White, A. (2003). “The Sourcebook of Parallel Computing”. Morgan Kauffman Publishers. Elsevier Science.
4. Juhasz Z., Kacsuk P., Kranzlmuller D. (editors) (2004). “Distributed and Parallel Systems: Cluster and Grid Computing”. Springer; First Edition.
5. Di Stefano, M. (2005). “Distributed data management for Grid Computing”. John Wiley & Sons Inc.
6. Miller, M. (2008). “Cloud Computing: Web-Based applications that change the way you work and collaborate online”. QUE Publishing.
7. Mc Cool, M. (2007). “Programming models for scalable multicore programming”. http://www.hpcwire.com/features/17902939.html
8. Chai, L., Gao, Q., Panda, D. K. (2007). “Understanding the impact of multi-core architecture in cluster computing: A case study with Intel Dual-Core System”. IEEE International Symposium on Cluster Computing and the Grid 2007 (CCGRID 2007), 471-478.
9. De Giusti, L., Chichizola, F., Naiouf, M., De Giusti, A., Luque, E. (2010). “Automatic Mapping Tasks to Cores - Evaluating AMTHA Algorithm in Multicore Architectures”. IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 2, No 1. 2010.
10. Olszewski, M., Ansel, J., Amarasinghe, S. (2009). “Kendo: Efficient Determistic Multithreading in Software”. Architectural Support for Programming Languages and Operating Systems (ASPLOS ‘09).
11. Bertogna, M., Grosclaude, E., Naiouf, M., De Giusti, A., Luque, E. (2008). “Dynamic on Demand Virtual Clusters in Grids”. 3rd Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC 08). Spain.

12. AMD, “Evolución de la tecnología de múltiple núcleo”. 2009. http://multicore.amd.com/es-ES/AMD-Multi-Core/resources/Technology-Evolution.
13. Burger T. W., “Intel Multi-Core Processors: Quick Reference Guide”. http://cachewww.intel.com/cd/00/00/23/19/231912_231912.pdf
14. Grid Computing and Distributed Systems (GRIDS) Laboratory-Department of Computer Science and Software Engineering (University of Melbourne), “Cluster and Grid Computing”. 2007. http://www.cs.mu.oz.au/678/.
15. Attwood, T. K., Parry-Smith, D. J. (2002). “Introducción a la Bioinformática”. Pearson Educación S.A.
16. Zhang, F., Qiao, X., Liu, Z. (2002). “A Parallel Smith-Waterman Algorithm Based on Divide and Conquer”. Proceeding of the Fifth International Conference on Algorithms and Architecture for Parallel Processing. HP, “HP BladeSystem”. http://h18004.www1.hp.com/products/blades/components/c-class.html.
17. HP, “HP BladeSystem c-Class architecture”. http://h20000.www2.hp.com/bc/docs/support/SupportManual/c00810839/c00810839.pdf.
18. Rucci, Enzo (2010). “Modelos de Comunicación en BLADE”. III-LIDI Technical Report.
19. Leopold, C. (2001). “Parallel and Distributed Computing. A survey of Models, Paradigms, and Approaches”. Wiley, New York.

Towards a High Performance Cellular Automata Programming Skeleton

A. MARCELA PRINTISTA[1,2] AND FERNANDO D. SAEZ[1]

[1] LIDIC- Universidad Nacional de San Luis,
[2] CONICET – CCT – San Luis.
Ejército de los Andes 950 (5700) San Luis. Argentina.
{mprinti, bfsaez}@unsl.edu.ar.

Abstract. *Cellular automata provide an abstract model of parallel computation that can be effectively used for modeling and simulation of complex phenomena and systems. In this paper, we start from a skeleton designed to facilitate faster D-dimensional cellular automata application development. The key for the use of the skeleton is to achieve an efficient implementation, irrespective of the application specific details. In the parallel implementation on a cluster was important to consider issues such as task and data decomposition. With multicore clusters, new problems have emerged. The increasing numbers of cores per node, caches and shared memory inside the nodes, has led to the formation of a new hierarchy of access to processors. In this paper, we described some optimizations to restructuring the prototype code and exposing an abstracted view of the multicore cluster to the high performance CA application developer. We introduce a new mapping strategy that can obtain benefit in the performance by adapting its communication pattern to the hardware affinities among processes allocated in different cores. We apply our approach to a two-dimensional application achieving sensible execution time reduction.*

Keywords. *Skeletal Programming, Cellular Automata, Multicore Nodes, Mapping Strategy.*

1. Introduction

Traditionally, parallel programs are designed using low-level message passing libraries, such as PVM or MPI. Message passing provides the two key aspects of parallel programming: (1) synchronization of processes and (2) communications between processes.
However, users still encountered difficulties because these interfaces force to deal with low-level details, and their functions are too complicated to use for a nonexpert parallel programmer. Many attempts have been undertaken to hide parallelism behind some kind of abstraction in order to free the programmer from the burden of dealing with low issues.
A alternative is to provide a set of high-level abstractions which provides support for the mostly used parallel paradigms. A programming paradigm is a

class of algorithms that solve different problems but have the same control structure. Parallel programming paradigms usually encapsulate information about useful data and communication patterns, and an interesting idea is to provide such abstractions in the form of programming templates or skeletons [6]. The essence of the programming methodology based on skeletons is that all programs have a parallel component that implements a pattern or paradigm (provided by the skeletons) and a specific component of an application (in charge of the user). A parallel skeleton encapsulates the control and communication primitives of the application into a single abstraction and frees the user to consider low-level details involved in high performance computing.

Cellular automata are ideally suited for parallel computing and consequently, researchers from diverse fields require support to design and implement parallel cellular algorithms that are portable, efficient, and expressive.

Today, multicore processors are an option to become part for clusters of multiple sizes. These multicore processors contain multiple execution cores on the same chip, each of which can independently perform operations, thereby introducing a new level of parallelization to clusters. These kinds of processors have emerged as a feasible alternative to use in high performance computing. As we explain later, many new factors must be considered and studied to achieve the best performance.

The goal of this paper is to describe the optimization techniques applied to a *CA* skeleton that will be able to take advantage of a multicore environment. We will show a performance improvement not due to modifications of the MPI implementation itself but rather due to a relevant process placement.

There are a substantial amount of work in the fields of parallel CA, with work done in both generalized tools and tools dedicated to particular applications. Several high-level parallel programming systems like CAMEL [12], CAM [11], StarLogo [9], and CAPE [13] made possible development of parallel software abstracting from the parallel architecture on which programs run. Actually, CAMEL has its own programming language CARPET [10] that allows the programming of cellular automata algorithms.

The rest of the paper is organized as follows. Section 2 gives a background about cellular automata and cellular algorithms. Section 3 discusses the CA skeleton and presents some experimental results of the CA implementation on a Cluster. Section 4 describes the multicore environments and discusses the MPI Process placement problem. We then introduce a new mapping strategy. Section 5 presents the comparative analysis of strategies and the conclusions are given. Finally, the future work is outlined en Section 6.

2. Cellular Automata Background

Cellular automata are simple mathematical idealizations of natural systems. They consist of a D-dimensional lattice of cells of uniform size connected with a particular geometry, and where each cell can be in one of a finite number of states. The values of the cells evolve in discrete time steps according to deterministic rules that specify the value of each cell in terms of the values of neighboring cells and previous values. Formally, a cellular automaton is defined as a 4-tuple (L_D, S, V, f) where L_D is a D-dimensional lattice partitioned into cells, S is a finite set of states $(|S| = v)$, V is a finite set of neighborhood indexes, and $\delta: S^v \rightarrow S$ is a transition function.

Below we summarize the most important characteristics that define the behavior of CA:

1. Initial State: The initial configuration determines the dimensions of the lattice, the geometry of the lattice, and the state of each cell at the initial stage.
2. State: The basic element of *CA* is the cell. Each cell in the regular spatial lattice, can take any of a finite number of discrete state values. In the simplest case, each cell can have the value 0 or 1. In more complex case, the cells can have more different values. (It is even thinkable, that each cell has a complex structure with multiples values.)
3. Neighborhood: For each cell of the automaton, there are a set of cells called neighborhood (usually including the cell itself). A characteristic of *CA* is that all cells have the same neighborhood structure; even the cells at the boundary of a lattice have neighboring cells that could be outside the domain. Traditionally, border cells are assumed to be connected to the cells on the opposite boundary (that is, for one dimension, the right most cell is the neighbor of the left most one and vice versa). Other types of boundary conditions may be modeled by using preset values of the cell values for the boundary nodes or writing unique update rules for the cells at the boundary. The neighborhood structure and its boundary condition depend on the application under consideration. The most common neighborhoods are called Von Neumann and Moore.
4. Transition function: The set of rules that define how the state of each cell changes on the basis of its current state and the states of its neighbor cells. In a standard CA, all cells are updated synchronously.

2.1 Overview of Cellular Algorithms

From a computational point of view, *CA* are basically a computer algorithm that is discrete in space and time and operates on a lattice of cells. The Fig. 1 shows a simple algorithm that solves a two-dimensional generic cellular automaton. The algorithm takes as input a two-dimensional lattice of (*N x N*) and initializes the structure with some initial configuration. The simulation involves an iterative relaxation process.

```
AutoCel(Lattice,steps)
init(Lattice)
  for t = 1 to steps
   for i = 1 to N
     for j = 1 to N
        nextState(Lattice,i,j)
```

Fig. 1. Sequential CA approach

This process is represented in the algorithm with a iteration of steps *steps*. In each time step t, the algorithm updates each cell in the lattice. The next state of an element $s^{t+1}(i,j)$ is a function of its current state and the values of its neighbors. The relaxation process ends after *steps* iterations.
Cellular automata parallel systems allow to user exploit the inherent parallelism of cellular automata to support the efficient simulation of complex systems that can be modeled by a very large number of simple elements with local interaction only. In fact, it is possible to exploit the data parallelism intrinsic to the *CA* programming model coming from the possibility to execute the transition function on different sublattices due to the local nature of cell interactions. Therefore, multicomputer is the appropriate computing platform for the execution of *CA* models when real problems must be solved. In this approach, the update of cells is synchronized and executed simultaneously by all processing nodes. If a cell and its neighbors are in the same node, the update is easy. On the other hand, when nodes want to update the border cells, they must request the values of the neighboring cells on other nodes. A common solution to this problem is to let two neighboring lattices overlap by one row or column vector. After a node updates its interior elements, it exchanges a pair of vectors with each of the adjacent nodes. The overlapping vectors are kept in the boundary elements of the sublattices. If a neighboring node does not exist, a local boundary vector holds the corresponding boundary elements of the entire lattice.

3. High Performance Simulation for CA Models

Libraries, like PVM or MPI (low-level abstract models) give the programmer control over the decomposition task and the management of communication/synchronization among the parallel processes. Although MPI does not include explicit mapping primitives, most of its implementations have a static programming style.
In this case, MPI support can avoid the mapping and scheduling problems, however, the programmer's task becomes more complex. Its unstructured programming model based on explicit, individual communications among processors is notoriously complicated and error-prone.
To reduce this limitation and to increase the level of abstraction without lowering the performance, an approach exists to restrict the form in which the parallel computation can be expressed. This can be done at different abstraction levels. The model provides programming constructs: skeletons, that directly they correspond with frequent parallel patterns. The programmer expresses parallelism using a set of basic predefined forms with solution to the mapping and restructuring problems.
Following this approach, Saez et al. [4] implemented a versatile cellular automata skeleton and an environment for its use. The skeleton is written in C and MPI and is accessed through a call to the constructor CA_Call and its parameters list allows substantial flexibility, which will bring benefits in

different application domains. The skeleton enables us to write *CA* algorithms in an easy way, hiding parallel programming difficulties while supporting high performance.
The cellular space of the automaton is represented by an array of *D* dimensions (*D*-lattice), which contains N^D objects called cells. Inside of a cluster based on distributed memory system, the parallel execution using *P* processors (denoted p_0, p_1, .., p_{P-1}) is performed by applying the transition function simultaneously to *P* sublattices in a SPMD way. As a first task of implementation, it is necessary to find a division criterion to provide *P* sublattices of the automaton. The underlying idea for the implementation of lattice division functions is the establishment of a relation among *P* processors. The structure of divisions produced by the proposed scheme and the partnership relation established among processors give place to communication patterns that are topologically similar to a *Mesh*. This partnership is the responsible of the assimilation the communicational topology of a *CA*.

If $\sqrt[D]{P}$ is a natural number and *N* is multiple of *P*, then a *D*-lattice can be divided in *P* sublattices given place to a *D* dimensional Mesh (*D-Mesh*). For a *D*-lattice, meshes of different dimensions can be made (for example, an tree-dimensional lattice can be divided into sublattices of two or one dimension), but these are not relevant to the facts discussed in this paper. *P* determines the number of divisions produced on the lattice, *D* defines the dimension of the Mesh of processors and $\sqrt[D]{P}$ is the degree of each dimension. The skeleton implementation assigns each sublattice to a processor and let the nodes update them simultaneously. Independently of the topology, each processor will be responsible of *D*-sublattice $\left(\frac{N}{\sqrt[D]{P}} x \dots x \frac{N}{\sqrt[D]{P}}\right)$, which means the evolution of $\frac{N^D}{P}$ cells.
No matter what the simulation problem is attacked, a cell changes its current value through a set of rules that define its next state depending on its current value and the value of its neighboring cells. The rules are described by a function built by the user. At the end of each step, after update all cells in the local sublattice, all processors interchange the updated border cells and a communication among its neighboring processors in a *D*-Mesh topology takes place.
To improve performance, the implementation uses asynchronous (non-blocking) communication calls available in MPI. Once the data communication is issued, the process can perform the computations of those cells that do not depend on the data expected from the neighboring processors in the *D*-Mesh. Finally, the process waits for the end of the communication. The iterative process is repeated as necessary.

3.1 Experimental Results of the CA Implementation on a Cluster

In this section, we present some results obtained by using the skeleton prototype previously described, which does not consider multicore facilities. As a starting point, we propose to evaluate the performance of the described implementation, and then compare it with the approaches proposed in the next sections.
The cluster used for the experiments was a 32 Node IBM 3550, which is equipped with two Dual-Core Intel(R) Xeon(R) 3.00GHz per node. Each node has 4MB L2 (2x2). The nodes are interconnected by a Gigabit Switch.
The CA_Call prototype was applied to resolve the numerical solution of Laplace's equation by lattice relaxation, which is representative of the class of two-dimensional CA models we study. The problem considers Von Neumann neighborhood, which comprises the four cells (North/South/ East/West) orthogonally surrounding a central cell on a two-dimensional lattice. The processes were distributed in a round robin fashion among the 32 quad-core nodes of the cluster. The Figure 2 shows the experiments called nx1c. The references Mesh 2x2 and Mesh 4x4 correspond to 4 and 16 nodes, respectively. All nodes are usable by the MPI processes with the restriction that only a single MPI process runs on a given node. With this scheme, the operating system chooses on which core of the node a process is executed.

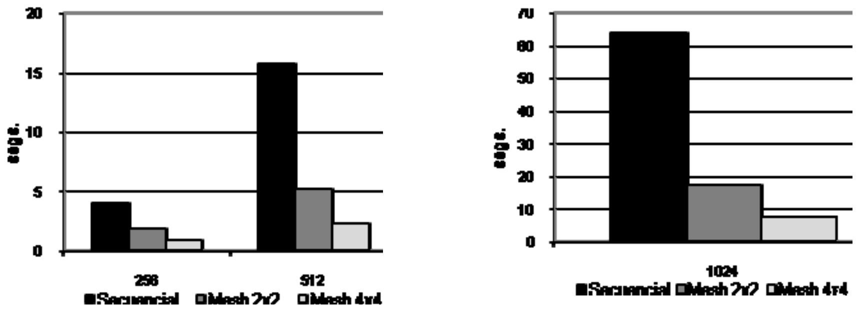
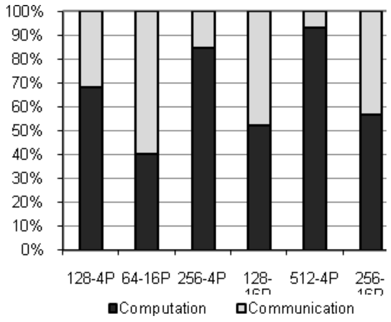
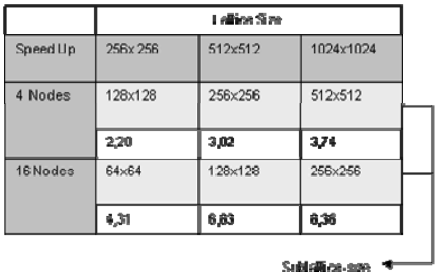


Fig. 2. Execution times on a cluster of 32 multicore nodes. One core by node.

In addition to overall execution time, we measured the computation and communication times for each lattice size. The Fig. 3 (left) shows that using 4 processors (Mesh 2x2), the computation ratios are much higher than the communication, obtaining, for example, in the case of an automaton of one million of cells a ratio of more than 90% of time involved in computation. As the lattice size increases, a speedup close to ideal can be visualized in the Fig. 3 (right). This configuration achieves to balance the degree of partitioning, the size of the problem and the communications involved. While for the same partitioning scheme running on 16 processors (Mesh 4x4), the relation between computing and communications begins to match. When this happens, the speedup is limited by the cost of communications and network latencies. This fact, give us some possibilities for the development of the proposals presented below.





	Lattice Size		
Speed Up	256x 256	512x512	1024x1024
4 Nodes	128x128	256x256	512x512
	2,20	**3,02**	**3,74**
16 Nodes	64x64	128x128	256x256
	4,31	**6,63**	**8,36**

Sublattice size

Fig. 3. Computation-Communication relation. Speedup.

4. Multicore Environments

Multicore processors have emerged and are currently the mainstream of general purpose computing. Quadcore processors are currently commonplace and core count by chip is expected to increase drastically in the forthcoming years. In HPC, these processors have been used as building blocks for cluster of multiple sizes, by grouping together a variable number of nodes (each containing a few multicore processors) through a commodity interconnection fabric such as Gigabit Ethernet.

A real challenge for parallel applications is to exploit such architecture at their potential. In order to achieve the best performance, many new factors must be considered and studied.

MPI programming model implicitly assumes the message passing as interprocess communication mechanism, so any existing MPI code can be employed without changes for running in a multicore cluster. From a point of view of programmers, pure MPI ignores the fact that cores inside a single node work on shared memory.

Moreover, it is not required for the MPI library and underlying software layers to support multi-threaded applications, which simplifies implementation.

But, the technology in study involves considering the different kind of communications among processes, depending on whether they are running on different cores within the same node or different nodes. Chai et al. [5] presented communication schemes on multicore clusters, where intra-chip, intra-node and inter-node are described. The speed of communication among cores in a multicore processor chip (intra-chip) varies with core selection, since some cores in a processor chip share certain levels of cache and others

do not. Consequently, intra-chip interprocess communication can be faster if the processes are running in cores with shared caches than otherwise. This asymmetry in communication speed may be worse among cores on distinct processor chips in a cluster node (intra-node) and is certainly worst if communicating cores belong to distinct nodes of a cluster (inter-node).
As an alternative to the pure MPI model, Rabenseifner et al. [8] presented the available programming models on hybrid/hierarchical parallel platform. The authors outline that to seem natural to employ a hybrid programming model which uses OpenMP [1] for parallelization inside the node and MPI for message passing between nodes. It can expect hybrid models to have positive effects on parallel performance. However, mismatch problems arise because the main issue with getting good performance on hybrid architectures is that none of the common programming models fits optimally to the hierarchical hardware.
Fortunately, recent MPI-2 implementations such as Open MPI [2] or MPICH2 [7] are able to take advantage of multicore environment and offer a very satisfactory performance level on multicore architectures. In particular, MPICH2 library is able to use shortcuts via shared memory in this case, choosing ways of communication that effectively use shared caches, hardware assists for global operations, and the like.
In the following sections, we describe some modifications to the *CA* implementation given in section 3 in order to restructure the prototype code and exposing an abstracted view of the multicore cluster to the *CA* applications developer.

4.1 MPI Process placement

In a multicore cluster based on distributed memory system, the parallel execution of *P* tasks, can be carried out by using several combinations between nodes and cores per node. Considering a homogeneous hardware environment -with a maximum number of nodes *M* and the same number of cores per node *C*- the node-core combination is composed by: $P = n * c$ where $1 \leq n \leq M$ and $1 \leq c \leq C$. This fact involves taking a decision about what node-core combination delivers the best performance, through the evaluation of the key features of the algorithm that can affect –positive or negatively- the expected performance.
According to the previous issue, for the *CA* model implementation can to exploit the underlying hardware, the MPI processes have to be placed carefully on the cores of the multicore cluster. Whilst MPI standard is architecture-independent, it is responsibility of the each implementation to bridge the gap between the hardware performance and the applications.
The next experiments were carried out linking the skeleton to MPICH2. The Fig. 4 shows the performance of *2-CA* for lattices of 256x256, 512x512 and 1024x1024 cells when the degree of partitioning applied to each lattice was 4, 16 and 64. In this experimental case all cores of assigned nodes are usable by the MPI processes with the restriction that only a single MPI process runs

on a given core. The experiments consider two different MPI process launching policies. The first policy uses the four cores per node in base to a simple sequential ranking. These experiments are called:

- `1nx4c`, to represent four processes on a 2-mesh of processors (2x2)
- `4nx4c`, to represent sixteen processes on a 2-mesh of processors (4x4)
- `16nx4c`, to represent sixty-four processes on a 2-mesh of processors (8x8)

The last experiment, `32nx2c`, represents sixty-four processes on a *2-mesh* of processors (8x8) using a round robin ranking. In this case, the mesh was made up of two cores from each of the 32 available nodes in the cluster.
In the case of sequential placement policy, we observed that as the degree of parallelism grows, the performance improves. However, the better performance is achieved when we do not fully use all the cores in a node (32 nodes under-subscribed). This configuration effectively provides more memory bandwidth to each core and improves the network latency experienced by each core, but it is not recommended because running with fewer than the maximum number of cores per node reduces overall throughput of a computing cluster. Besides the problem to assign processes to nodes, it comes other problem related to the distribution of processes to specific cores inside a single node. We observed that the default policy used by MPI not all distributions are able to establish automatically an affinity mechanism between processes and cores.

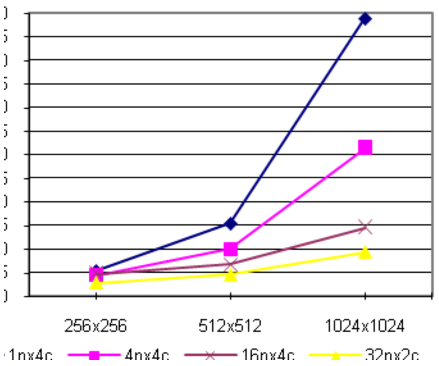


Fig. 4. Multicore Execution Times

On Linux Operating System, the system call `sched_setaffinity` has the ability to specify in which core within the node a certain process will execute. We incorporate in the skeleton this facility based on the knowledge of the hierarchy of multicore cluster.

The Fig. 5 shows the execution time of the *CA* skeleton as a function of lattice size, applying an explicit affinity between those neighboring processes that exchange data and that have been allocated in the same node.
As expected, there are vast reductions in the execution times, showing an average reduction of the order of 13%, 25% and 30% for `4c-Aff, 16c-Aff` and `64c-Aff` experiments respectively.

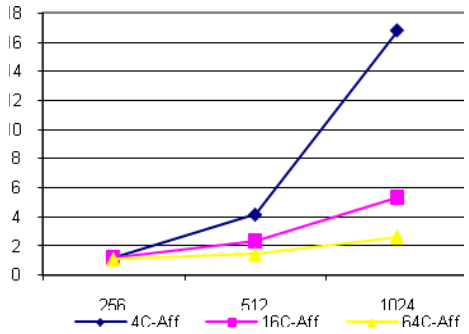


Fig. 5. Multicore Execution Times using Affinity

It is important to realize that the *CA* skeleton was designed to allocate, in matrix notation, the sublattice *(ij)* to the MPI process with rank $j + i * P^{1/D}$. In sequential order, ranks 0..3 go to the first node, ranks 4..7 to second node, and so forth. In round robin order, the MPI process rank 0 goes to the first node, rank 1 to the second node, and so forth. In any case, the *CA* model topology maps efficiently to the hardware topology. This leads to design a new policy to distributing sublattices to MPI process, which is explained in the next section.

4.2 The Mapping Problem

By applying different strategies in the prototype, either in the skeleton code (e.g., affinity) as in its execution environment (MPICH2), the skeleton implementation has achieved a considerable reduction execution time.
No matter of use case, a relevant observation is that the parallel implementation of the *CA* model, includes stable communication patterns in which data interchange occurs among neighboring sublattices. However, the methodology of allocating work to the MPI processes does not regroup sublattices on the same node as much as possible in a way that it reflects the behavior based on neighborhood of the *CA* model. This can be observed graphically in Fig. 6 (left), for a *2-Mesh* (8x8). All nodes have the same setup, for example, sublattices 00, 01, 02 and 03 are assigned to node 0, sublattices 04, 05, 06 and 07 to node 1 (not showed in the figure) and

sublattices 10, 11, 12 and 13 to node 2. Each node must manage ten inter-node communications (marked with deep blue), two intra-node (dark blue) and four inter-chip (light blue).
The Fig. 6 (right) shows the configuration when the *CA* model implementation applies a new mapping of sublattices to the different processes. This new mapping accomplishes two objectives: (1) all nodes manage the same amount of inter-node communication and (2) it takes advantage of multicore nodes hierarchy. As can be seen in the figure, of the four neighbors of a sublattice, one of them is mapped on the same chip with which it shares the cache, the other is mapped on the same node with which it shares the memory and the interchanges with the other two neighbors inevitably require inter-node communication.

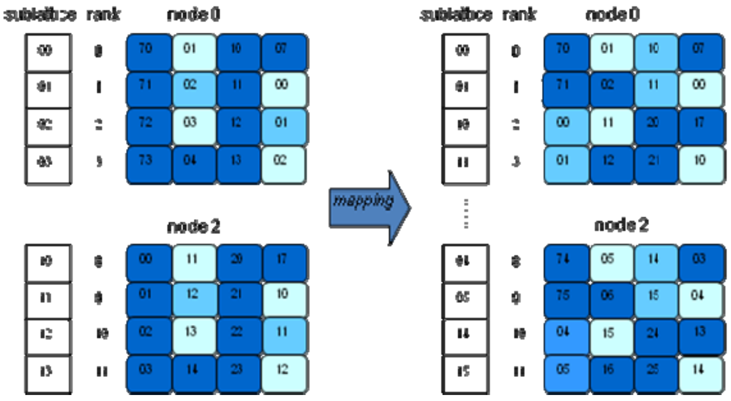


Fig. 6. Mapping of sublattices to MPI processes ranks in a 2-Mesh (8x8)

5. Results and Final Remaks

In a previous work, we have presented an implementation of parallel *CA* skeleton. It attacks the classical problems inherent to parallel programming such as task and data decomposition and communications. The skeleton frees the non-expert user from the burden of dealing with low issues of parallelism. The emergence of multicore processors with shared caches and non uniform memory access causes the hardware topology to became more complex. Therefore, a real challenge for parallel applications is to exploit such architecture at their potential. In order to achieve the best performance, many new factors must be considered and studied.
In this paper, we carried out experimental work that enabled us to understand the behavior of the architecture and implementation of the skeleton. This was possible because the parallel *CA* model is a typical application in which the

data interchange occurs among neighbor sublattices and the communication pattern does not change across multiple executions.

The first experiment scattered the processes among the nodes. The second experiment regrouped the MPI processes on the nodes, but the operating system chose the placement of them among the cores. The third experiment also regrouped processes, but the skeleton implementation applied an affinity based on the knowledge of the hierarchy of multicore cluster. These experiments showed how the different MPI processes launching strategies impact in the performance on a multicore cluster and they were useful for exploring the hierarchy of the architecture.

Afterwards, we show a new mapping strategy that can obtain benefit in the performance by adapting its communication pattern to the hardware affinities among processes. Figs. 7 and 8 show a comparison of the implemented strategies in this work considering tree different lattice sizes. For the experiments called `-Aff` and `c-Aff-Mpp`, the MPI processes were allocated on the same node as much as possible, i.e. in sequential order.

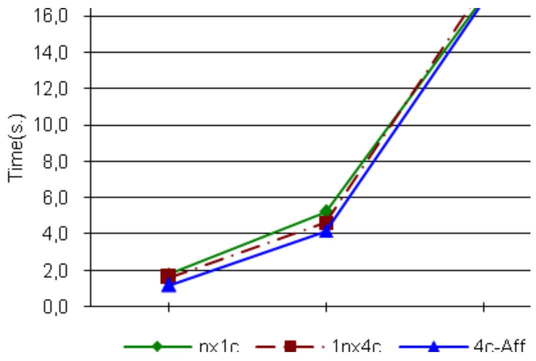


Fig. 7. Execution Time using 4 cores

For a small mesh size as 2x2 (partitioning degree=4), the multicore strategies performed in much the same way as the non multicore one (`nx1c`). This behavior illustrates how the high costs of sequential computation in each core can not support optimizations. For 16 and 64 cores, the simultaneous application of all multicore strategies are the cases that achieve better performance.

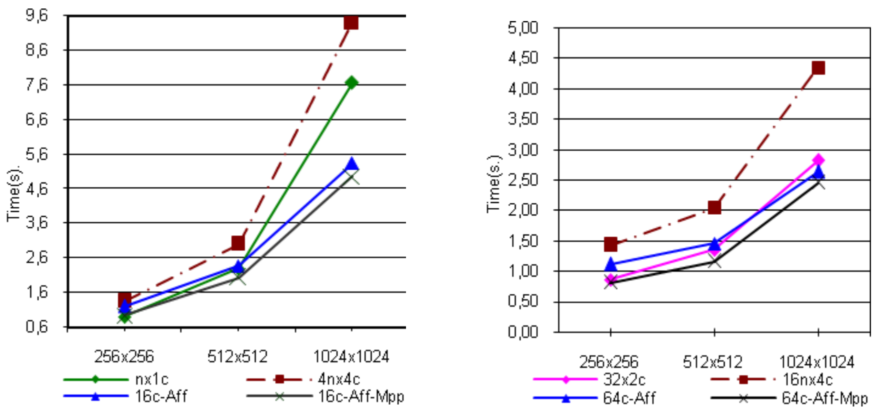


Fig. 8. Execution Time using 16 (left) and 64 cores (right)

6. Future Work

We have implemented a first version of the high performance *2-CA* skeleton and early experiences showed us that the strategy of allocation is very important in a multicore environment. But, all developments were performed on a particular type of cluster, of quadcore nodes. We are working to generalize the mapping strategy to cluster with larger number of core per node, where the hierarchy of memory access can be even greater. We are also examining other factors that probably influence the performance of communications, as cache and shared memory sizes.

Acknowledgments

We wish to thank the Department of Computer Architecture and Operating Systems of the Universidad Autónoma de Barcelona for allowing us to use their resources. Also to the Universidad Nacional de San Luis, the ANPCYT and the CONICET from which we receive continuous support.

References

1. OpenMP architecture processing reference model. ITU-TX.901,ISO/IEC 10746-1. available at http://enterprise.shl.com/RM-ODP/default.html.
2. Open MPI: Open Source High Performance Computing. http://www.openmpi.org.

3. Norman, M.G., Henderson, J.R., Main, G. and Wallace, D.J. (1991). The use of the CAPE Enviroment in the simulation of Rock Fracturing. Concurrency: Practice and Experience 3, 687.
4. Saez, F. and Printista, M. (2009). Parallel Cellular Computing Model. Proceedings of the IADIS international conference,Vol 2, 145-149.
5. Hartono, A., Chai, L. and Panda, D. K. (2006). Designing High Performance and Scalable MPI Intra-node Communication Support for Clusters. The IEEE International Conference on Cluster Computing (Cluster 2006).
6. Cole, M.I. (1989). Algorithmic Skeletons: Structured Management of Parallel Computation. Research Monographs in Parallel and Distributed Computing. Pitman, London, UK.
7. MPICH2. http://www.mcs.anl.gov/mpi/.
8. Jost, G., Rabenseifner, R. and Hager, G. (2009). Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes. In Proceedings of the 17th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2009), 427436, Weimar, Germany.
9. Resnick, M. (1994). Turtles, Termites and Traffics Jams. The MIT Press.
10 Spezzano, D., Talia, G. (1996). CARPET: A Programming Languaje for Parallel Cellular Processing. In Proc. 2nd European School on Parallel Programming Environments (ESPPE 96).
11. Margolus, N. and Toffoli, T. (1987). Cellular Automata Machines A New Enviroment for Modeling. The MIT Press. Cambridge, Mass.
12. Spezzano, G., Talia, D., Di Gregorio, S., Rongo, R. and Spataro, W. (1996). A Parallel Cellular Tool for Interactive Modeling and Simulation. IEEE Computational Science & Enginieering 3, 33.

IX

Information Technology Applied to Education Workshop

Voice command adaptation for Jclic, for the special education context

M. LUCRECIA MORALEJO[1,3], STEFANIA OSTERMANN[2], CECILIA V. SANZ[3] AND PESADO PATRICIA[1,3]

[1] III LIDI, School of Computer Science. Universidad Nacional de La Plata, Argentina.
[2] School of Computer Science. Universidad Nacional de La Plata, Argentina.
[3] CIC (Comisión de Investigaciones Científicas), Buenos Aires, Argentina.

Abstract. *In this paper, the adaptation of an educational software application with voice commands for students with motor disability who have no speech impairments is proposed. As part of this process, some educational program and adaptive software applications were analyzed. Integration tests with several adaptive software applications studied with JClic were also carried out to analyze the assistance they can provide students with some type of motor disability in activity solving. Different voice recognition (VR) motors were studied, as well as their theoretical basis. The analysis of the VR motor Sphinx-4 was detailed, studying the design architecture and development of the selected educational tool, JClic. Finally, the development of a prototype with the adaptation of JClic was carried out, with the integration of Sphinx-4 to provide VR, in particular, for simple association activities.*

Keywords. *ICTs, educational software, motor disability, voice recognition, technical aides.*

1. Introduction

Nowadays, there is a large number of software oriented to education on its various levels. Many of these have been accepted or created taking into account student diversity, but others are just standard tools that do not offer any adaptation, which means they are targeted to a restricted set of students [1]. People affected by some kind of motor disability usually have difficulty in some basic skills related to perception (visual, auditory and tactile), communication, movement and/or handling. As a consequence, they face numerous obstacles and barriers that prevent them from developing skills, carrying out activities, relating to other people and the environment, etc. For people with special needs, the mere use of ICTs may represent the achievement of a high degree of autonomy in their personal lives [2].
One of the reasons for the limited deployment of ICTs in special education is the diversity and specificity of needs. Their use as tools, in this field, requires very complex or varied developments, some customized, that are going to be used by not-very-numerous groups.

The current situation presents great challenges to overcome for a disabled person to be in a position of equality with the rest of the population. Therefore, the environment has to be appropriately adapted and technical aides should be used that allow a maximum elimination of the barriers that prevent disabled people to interact with their environment. However, this has been the main conflicting issue: people with motor disabilities usually do not have the technical aides and adaptations they need to interact with a hostile environment [3].
These are the reasons behind the development of adaptations to a software application that is very widely used in the educational context, such as JClic, to facilitate its use by students with motor disabilities and thus encouraging their intellectual development.
Even though there is a known relation between motor disability and speech development difficulties, this is not so in every case. This paper is aimed at people with motor problems, but with little or no consequences in language development. This subset of people was selected because there is a wider variety of technical aides for people with motor disability that use various parts of the body, and we considered that the use of voice would be a good alternative if the person affected by the disability had no difficulty in oral expression. Also, this type of adaptation would require less effort from the person to use the computer, which would help prevent injuries caused by a “repetitive strain”.

2. Selection of Jclic as tool to be adapted

In the market, there is a large number of educational software applications available. With the purpose of having a diverse panorama of available applications, an analysis was carried out with those which offer various functionalities and could be used on different educational levels. Among those, JClic, Textoys, Hot Potatoes, Markin, Lim, and Wink were included.
Among these applications that were analyzed, the JClic tool, which is an environment for creating, carrying out and assessing multimedia educational activities, was selected. It is formed by three main components – JClicAuthor, used for creating activities, JClicPlayer, used to solve activities, and JClicReports, used to compile data from solved activities. This application is used to carry out various types of educational activities: puzzles, associations, text exercises, crosswords, etc. [4]. In general, activities are not presented alone, but packed into projects. A project is formed by a set of activities and one or more sequences that indicate the order in which they are to be shown.
Some of the features that led to the selection of this software application were that it is developed under a GPL license, which means that the source code of the program is available for study and analysis. Thus, it was possible to carry out the integration proposed. It is also one of the most widely used software applications for carrying out educational activities (it has been used in the

educational context for years now), so it was considered that it would be interesting to develop a prototype to widen user diversity for these activities.
A second strong reason for this selection is that JClic can be used on various operating systems, such as Windows, Linux, Solaris and Mac OS X. This is because JClic was entirely developed with Java technology, which is multiplatform.

3. Voice recognition

Technological advances have provided human beings new and greater possibilities of developing a fuller lifestyle, but at the same time, this lifestyle continuously demands new and specific knowledge and skills for individuals to be able to take advantage of the possibilities being offered. In the case of people with some type of disability, the progressive complexity of the social media may however have the opposite effect to the desired social progress [5].
Thus, voice recognition is an alternative to communicate with computers, allowing people with motor disabilities who cannot access the standard keyboard and/or mouse to, through speech, perform actions that would not be possible for them without this technology; in other words, the purpose is to convert human speech into actions that the computer can interpret.
This technology is a part of Artificial Intelligence whose purpose is to allow voice communication between human beings and electronic computers, i.e., the process of converting a spoken message into text that allows the user to communicate with the computer. The problem to solve with any VR system is that of achieving the cooperation of a set of data from various knowledge sources (acoustics, phonetics, phonology, lexicography, syntax, semantics and pragmatics) in the presence of inevitable ambiguities, uncertainties and errors to arrive at an acceptable interpretation of the acoustic message received [6]. A voice recognition system is a computational tool that is able to process voice signals issued by human beings and recognize the information they contain and convert it into text or issue orders that act on a process [7]. Various disciplines are involved in its development, such as: physiology, acoustics, signal processing, artificial intelligence, and computer science.
There are some greatly significant components for VR systems: the dictionary, grammar, the acoustic model, and the language model. The dictionary represents the set of words or sounds to be recognized. Unlike a standard dictionary, each input is not necessarily a single word; it can be as long as a sentence or two. The smallest vocabularies may include one or two sounds to be recognized, whereas very large vocabularies may have hundreds of thousands or more. Grammar is defined based on the words that have to be accepted by the application, and can be given through a style that is similar to the BNF.
The language model can be tackled through statistical models (Statistical Model Language - SLM) or by using finite state grammars (FSG) [8]. A statistical model captures word and word sequence probability. It is used in the decoder to limit the search and, in general, makes a significant contribution to recognition accuracy. A good model is that which accurately

models the expected input. It is characterized by its order, in terms of "n-gram", where "n" indicates the size of the window over which statistics are computed. In general, the larger "n", the more accurate the model. Also, as "n" increases, more data are required to ensure a correct estimation of statistics. A finite state grammar defines possible words, as well as their possible orders.

An acoustic model is created from recordings, their respective transcriptions, and the use of software to create statistical representations of the sounds forming each word. The performance of the recognition achieved by the acoustic model can be further improved by means of a language model, which helps avoid the ambiguity among several similar words produced by the acoustic model.

For the selection of the tool to be used, various voice recognition software applications were analyzed, including Loquendo, Xvoice, NicoToolkit, Sphinx, and Dragon Naturally Speaking. Their main features, functionalities and requirements were studied.

Among the applications analyzed, Sphinx in its version 4 was selected. It is a system developed at Carnegie Mellon University (CMU) [9]. This framework is a system based on hidden Markov models (HMM) so, as a first step for its operation, it must first learn the characteristics (or parameters) of a set of sound units, and then use the knowledge acquired from these units to find the most likely sequence of sound units for a given voice signal. This particular tool was selected because it is widely used by researchers and developers working in the area of voice recognition and, therefore, it is constantly developed and updated.

Due to its licensing characteristics, it can be freely used for any development and research activities. Also, its source code can be obtained, in case any modification were required or its low level operation were to be studied. It is completely developed with Java technology, the same as JClic. Thus, it served the purpose of integrating both components without the problems of language incompatibility. Additionally, it has been designed with a high degree of flexibility and modularity, where each system element can be easily replaced or modified. It is through the Configuration Manager that the framework provides the possibility of dynamically loading and configuring the various modules, during runtime. Thus, the components that are going to be used, and their specific configuration, are determined. In particular, the dictionary and grammar to be used during recognition can be specified. Below, the specific proposal for this work is presented.

4. Adaptation proposal

The adaptation proposed has tackled the modification of JClic activities so that they can be solved through the use of voice commands. To this end, the simple association type of activity was initially considered.

In this type of activity that can be created in JClic, the user has to discover the relations that exist between two sets of information. That is, two groups

of data with the same number of elements are presented, where each element in the source data set corresponds to an element in the target set. This one-to-one relation is what makes this a simple association, in contrast with complex associations, where each source element may correspond to 0, 1, or more target elements.
As a first step to carry out this integration, some decisions had to be made, as detailed below.

4.1. Stage 1: Analysis

One of the decisions considered was how to inform that the activity will be done using voice commands.
It was considered that in this situation, the user needs the assistance of the teacher, since the latter is in charge of deciding if the use of VR is appropriate for each specific student. To do this, the program shows a prompt on the screen before starting the activity. It asks the user to indicate if voice recognition will be used.
Another important issue was deciding on the mechanism that should be provided to identify each interactive element on the screen, with the purpose of solving the activity. In this regard, various possibilities were analyzed. This identification used by the user to name an element will be called label from here on.
First, the possibility of using the letters of the alphabet as labels was considered, but this turned out to be impracticable due to the phonetic similarity between certain letters in Spanish, such as “b” and “d,” which considerably decreased recognition success rates.
On the other hand, if the number of checkboxes to be used increased, it was more natural to use combinations of digits (e.g., 10) than using combinations of letters (e.g., ab). Also, not all letters could be used; those that caused phonetic conflicts, such as the ones already mentioned, or those whose pronunciation was complex, such as the case of letter "r," had to be removed from the dictionary. This considered, the decision was made to use numbers for the creation of labels. This solution presents certain advantages in relation to the first option proposed.
Additionally, it was decided to include all necessary adaptations to avoid difficulties in the pronunciation of certain numbers. To do this, alternative pronunciations to the correct word were considered. For instance, users can say “tes” instead of “tres,” “tinco” instead of “cinco,” “acetar” instead of “aceptar,” among others.
Even though this decision results in a larger dictionary, it has positive consequences as regards the number of users for whom the prototype would be accessible. Thus, a balance between application performance and product usability was attempted.
The second issue that had to be solved was that of knowing when the user finishes naming the two elements to be joined. To do that, the use of “connecting” words was considered. For instance, “uno con tres aceptar” (one with three accept), which is interpreted as follows: the first number

("uno," one) represents a checkbox from the first set of data, the connecting word “con” (with) indicates that the user is about to name the checkbox from the second set, represented by the second number in the phrase ("cinco," five). The word “aceptar” (accept) confirms that the user wants to join both checkboxes.
Also, as regards labels, a decision had to be made on where to add the necessary code for the label to be inserted in the component representing the checkbox with the information. It should be mentioned that they are generated when JClicPlayer is run but only if the user indicates that voice commands will be used to carry out the activity. This involved a decision, since the presentation of the information from both sets had to remain random, so that the activity did not appear already solved because of the use of labels.
Finally, the necessary code was added in such a manner that the application shows a message prompting for confirmation of what the user said. Thus, when the user names the checkboxes to be joined, the program presents a message with the words that were recognized. To confirm the recognition, the user says “aceptar” (accept); otherwise, the word used is “cancelar” (cancel).
Below, the second stage of the work is presented, which was deciding on issues related to the VR motor and implementing those decisions.

4.2. Stage 2: Configuration of Sphinx-4

First, in order to use Sphinx, the application has to be downloaded from the official site [5]. The site also offers the source code of the tool for those who wish to make changes. If, however, as in our case, no modifications are to be introduced, all that has to be done is including the .jar file in the application where it will be integrated.
Currently, Sphinx-4 has models that have been created with SphinxTrain (training tool included), and it can be downloaded from the cmusphinx.org site.
At first, the idea of creating the dictionary using the WSJ_8gau_13dCep _16k_40mel_130Hz_6800Hz model that is included with the distribution of Sphinx-4 was considered as a valid alternative, by replacing English phonemes for those corresponding to Spanish. There are reviewed works in the VR area that perform this type of solution[1].
Even though phonemes correspond to the English language, during a first stage they were used to generate the dictionary for the integration with JClic.
This solution was partially valid, since the recognizer worked with a high success rate. However, two shortcomings were found. On the one hand, there were errors in recognizer accuracy in noisy environments. This would be a problem in those cases when the adaptations were to be used in schools, where there are several students in the same classroom. On the other hand, if

[1] Among these, the Mouse Advanced GNU Speech (Magnus) project was consulted: http://magnusproject.wordpress.com/

the dictionary were to be extended to use words with the letter “ñ,” there were no phonemes in English that could represent the corresponding sound.
Based on these conclusions, it was decided to switch to a model based on the Spanish language. After some research on the topic, two viable alternatives were found. One of the options was training the recognizer with the SphinxTrain tool, while the other was using models that had already been trained and tested. For this development, an already trained model was selected, but some tests were also done with the training tool in order to understand and study its operation.
To do this, an already trained model that was available on the Web and whose use was free, was used. The project is called Diálogos Inteligentes Multimodales en Español (DIME, Intelligent Multimodal Dialogues in Spanish), and it offers more than one acoustic model. The model selected for this work is called DIMEx30-T22 [10].
Using this list of phonetic units, the dictionary to be used in the integration with JClic was created. It would have been possible to incorporate the dictionary exactly as presented by DIMEx30, but there were some words that were missing, so it was redefined using the same original phonetic units. As regards the language model, the definition of the acoustic model and its architecture, they were used exactly as offered by DIMEx30.
To incorporate these files to JClic, first a .jar file had to be created which, by standard, should follow the directory structure of the models provided by Sphinx-4.
After creating the .jar file, it was included in the classpath of the application.
Also, Sphinx-4 had to be configured to incorporate the new acoustic model files, dictionary, grammar, and language model. This was done through the configuration file (Configuration Manager). In the following section, aspects related to the development of the prototype are detailed.

4.3. Stage 3: Prototype development

In this section, aspects of the prototype including both components used for the integration will be discussed. One of these is the procedure followed to incorporate the voice recognition framework into JClic. To do it, a class representing the recognizer was created in JClic, called VoiceRecognizer, where the main methods are included, such as the method used for its creation, as well as the method that is responsible for carrying out the recognition itself. A package called “recognition” was created within the “src” package of JClic. Then, this class is used in the builder method of the Player class if the user chooses to work using voice recognition. This is where the recognizer is created to start working.
Also, the class representing the recognizer was configured to inherit from SwingWorker, even if Swing is not used, so that JClic and the recognizer run on separate threads that interact with each other, so as to parallelize tasks. Thus, both components can be executed seamlessly.
To carry out the task of solving a simple association activity, upon its creation, the recognizer runs a method called getCommand() in the class

representing the corresponding activity. This method is responsible for processing the voice input from the user and making the corresponding decisions.
When an “aceptar” (accept) voice input is received, the system shows a message with the values that are going to be processed; the user has to confirm the values for the action to be performed.
For the confirmation, the word “aceptar” has to be uttered again. After confirmation, a method is invoked that is responsible for executing the action that the user wishes to perform. This method looks for the checkboxes that were named, if they exist and have not already been selected. Then, it checks within the internal structure of the elements if the correspondence is correct, i.e., if the cells selected are part of the solution. If so, the elements are removed from the set of possible elements to be chosen, and it moves on to the next correspondence, until getting to the last one. When the last correspondence is checked, the activity finishes.
JClic provides a module that can keep track of the time used in each activity, attempts, correct answers, etc. Even though time can vary if voice recognition is used, the counters for attempts and correct answers were kept unchanged to allow the teacher to evaluate student performance for the activity. For this reason, a message prompting the user to confirm the answer was added, since most recognizers introduce a certain error rate. This means that the recognizer could interpret a wrong answer and JClic would record it as a failed attempt on the part of the student, harming student performance evaluation. With these additions, the teacher can use the error counter provided by default by JClic.
The prototype developed so far includes, as already mentioned, the resolution of simple association activities. However, as part of this work, a proposal has been made for a possible strategy to extend the prototype to the remaining activities. This will be tackled in future works. In the following section, the integration strategies proposed so far are assessed.

5. Assessment and conclusion

The prototype described in this article was provided to experts in the field (working in the various areas involved in this work) for them to express their opinions.
This type of test was carried out first in order to analyze the results and take them into account for future lines of research work. After this stage, the prototype will be tested with end users, including both teachers and students. The reason to carry out this testing with the end users as a second stage was to avoid having students experience possible failure situations, typical of the software testing and strategy itself stages. Also, this methodology offers the advantages of producing quality feedback and a thorough analysis by the experts. Experts offer their thoughts about the object to be assessed. Through this expert opinion, it is expected to obtain reasonably good assessments and guesses in situations where no exact quantifications can be obtained, or doing

so is not advisable [11]. However, these assessments can, and should, be confirmed or modified in time, as information on the study object is collected.
A survey with close-ended and open-ended questions was used as assessment instrument to collect the information needed to asses the prototype.
The following results were obtained as a conclusion of the surveys answered by the experts: The selection of the educational software to be adapted was good, as well as the use of voice commands as technical aid. As one of the experts mentioned, this option can be used as a complement with other tools and is not necessarily better or worse than any other adaptation, but a different alternative that opens a road of new possibilities. Even though only a few experts commented on the selection of the voice recognition motor, those who did agreed that it was correct. The main aspects to be highlighted are its availability and possibilities as regard functionality. In the context of this work, it is considered that the use of Sphinx-4 was convenient, in agreement with the feedback received from the experts.
Finally, the solution strategy proposed was analyzed, and the experts expressed their agreement and offered some alternatives to take into account in future works. Some of these aspects are detailed in the following section.

6. Future lines of work

Even though a significant amount of knowledge has been collected on various tools, both educational and in relation to VR, there are still certain modifications, improvements and extensions to be developed in the adaptation presented, considering as well some suggestions analyzed after the assessment carried out.
The following future lines of work are proposed:

- Carrying out tests with students and teachers.
- Allowing label configuration (the teacher could choose to label each checkbox as desired)

One of the improvements that has already been implemented is how to analyze if voice commands will be used to solve the activity. Before, every time a project was loaded from JClicPlayer, the user was asked if the activity would be solved by means of voice commands. Since JClic is used beyond the context of special education, this prompt was oftentimes unnecessary, since no student would be using voice commands. In the cases of special education, it will be a decision made for each student. For this reason, this decision was initially the responsibility of the teacher. The teacher decides if, when the project is loaded, the prompt to use voice commands is shown or not.

Currently, the prototype is being extended to the remaining activities in JClic. The next type of activities to be added will be those of complex association and memory games.

References

1. Sánchez Montoya, R. (2007). "Capacidades visibles, tecnologías invisibles. Perspectivas y estudios de casos". Seminario Internacional Virtual: "Las nuevas tecnologías de la información y la comunicación aplicadas a las necesidades educativas especiales". Perú. http://www.ordenadorydiscapacidad.net/Capacidades.pdf
2. Castellano, Sacco, Zurueta (2003). "La utilización de software de uso general y aplicaciones específicas en el área de las discapacidades motrices". IV Congreso Iberoamericano de Informática en la Educación Especial. http://www.niee.ufrgs.br/eventos/CIIEE/2003/
3. Perez, F.J. y Rodríguez Vázquez, J. (2004). "Tecnología. Educación y diversidad: retos y realidades de la inclusión digital. Propuestas de futuro". 3º Congreso Nacional de Tecnología. Educación y Diversidad. Conclusiones. Biblioteca TECNONEET. http://www.tecnoneet.org/conclu04.php.
4. http://clic.xtec.cat/es/jclic/index.htm.
5. http://www.tecnoneet.org/docs/2002/2-82002.pdf.
6. Bernal Bermúdez, Bobadilla Sancho y Gómez Vilda (2000). "Reconocimiento de voz y fonética acústica". México, Alfaomega grupo Editor.
7. Rocha, Luis (1986). "Sistemas de reconocimiento de voz". Revista telegráfica electrónica. Agosto, 1172-1180.
8. http://sphinx.subwiki.com/sphinx/index.php/Language_model.
9. http://cmusphinx.sourceforge.net/sphinx4/
10. http://leibniz.iimas.unam.mx/~luis/DIME/recursos.html.
11. http://www.insht.es/InshtWeb/Contenidos/Documentacion/FichasTecnicas/NTP/Ficheros/401a500/ntp_401.pdf.

E-mail processing using data mining techniques

AUGUSTO VILLA MONTE, CÉSAR ESTREBOU AND LAURA LANZARINI

III-LIDI (Institute of Research in Computer Science LIDI).
Faculty of Computer Science, Universidad Nacional de La Plata, Argentina.
{avillamonte, cesarest, laural}@lidi.info.unlp.edu.ar.

Abstract*. A proposal to use data mining techniques to analyze e-mails corresponding to courses carried out through a distance education platform is made. The purpose of this type of analyses is determining which are the groups of relevant words that allow establishing communication topics of interest. Even though this new information can have various applications, they all involve an improvement in student service. The method proposed has been applied to the e-mails of the PACENI Project (Support Project for Improving First-Year Teaching in Courses of Studies in Exact and Natural Sciences, Economic Science and Computer Science) with satisfactory results.*

Keywords. *Information Retrieval, Text Mining, e-mails analysis, topic detection.*

1. Introduction

Distance education platforms are a learning environment through which teachers and students interact by performing various types of activities.
In this context, electronic mail is the most commonly used mechanism, and it is therefore of interest for the study of techniques that allow analyzing and modeling the information shared through this medium. For example, it would be relevant knowing the topics most frequently enquired by students. This could have various applications:

- It would allow detecting shortcomings in the information provided, for instance, lack of information regarding exam dates or the need for reinforcement in any given topic because the theoretical material provided has not been clear enough.
- Automatically organizing e-mails to improve student service.
- Automatically identifying core discussion topics in order to improve decision-making.

An e-mail has a date, a set of addresses, a subject, and a body. The latter, even though it may contain various types of information, consists basically of text and can therefore be analyzed by means of text mining techniques.
Text mining is a branch of Data Mining, and its main purpose is the extraction of high-quality information from documents.

It has numerous applications in various areas:

- In Biomedicine, it has been used to automate the identification and extraction of information from the numerous papers published each year [1].
- In Molecular Biology, it has been used to automatically extract information about genes, proteins and their functional relations from large collections of texts [2].
- In Education, it has been used to facilitate resource searches by combining the documents from various Web sites from related organizations [3].
- In the commercial context, it has been used to analyze the information generated by a consumer complaint Web site in order to obtain word relations that allow understanding the data [4],
- In the hospitality industry, using information available on Internet about hotels and possible tourists, it has been used to develop competitive strategies by analyzing demographic features and browsing habits [5].

All these works are representative of the diversity of areas in which text mining techniques are applied. However, regardless of the type of problem at hand, in most of the cases the main purpose is determining the relevance of the document based on a previous query. This allows more efficient automatic classification and access.
However, the extraction of information from e-mails is based on some special considerations, since, in general, the texts are short and their wording is quite abbreviated. Thus, some of the metrics used are no longer relevant, such as text length or the frequency of any given word within it.
The method proposed in this paper was applied to the e-mails of the Tutors Program (PACENI). This program is promoted by the Ministry of Education and its purpose is reducing the number of students that drop out from their university courses of study during their first year. This program was implemented at UNLP in the 2009 school year. Through it, first-year students are accompanied by tutors, post-graduate students or advanced students, who help them overcome the initial difficulties of university life.
For the processing stage, a dictionary built automatically from the reduction of each word to its root (stemming) [6] and its subsequent selection was used. By using this dictionary, each e-mail was represented as a numerical vector and was then used to train a SOM (self organizing map) neural network. From the weights of each neuron in the trained network, the most frequent combinations of terms can be identified. Finally, association-rule metrics are used to establish the relevance of each combination.
This paper is organized as follows: in Section 2, some related works are mentioned; in Section 3, SOM networks and their training mechanism are briefly described; in Section 4, the method proposed is detailed; in Section 5,

the results obtained are presented; and in Section 6, conclusions are drawn and future lines of work presented.

2. Related work

Obtaining information from e-mails is a relevant task whose main purpose is classification and interpretation.
In this sense, the identification of spam e-mails is a generalized problem and has therefore received a lot of attention [7, 8, 9, 10, 11].
There are also approached that seek to automatically identify the author of the e-mail or the core subject of the message. For example, [12] tries to identify the person writing the e-mail from features based on number of words, number of lines, and the frequency of significant key words. [13] proposed a method that assesses the words from e-mails based on their age. The age of a word is calculated based on the frequency with which e-mails including it are received. The problem of this approach is the number of different words that can be used to refer to the same concept.
There is a current approach that has become popular with the appearance of various social networks in work environments. Nowadays, the development of collaborative tasks and the use of e-mails as communication mechanism are common. This creates the need of solving some participation-related issues, which implies identifying project members and their categories, as well as central work topics [14].
The general objective of this paper is related to this latter approach—we try to obtain information from a group of e-mails generated by teacher-student relations during a course carried out through a distance-education platform.

3. SOM (Self-Organizing Maps)

The SOM (Self Organizing Maps) neural network was defined by Kohonen in 1982 [16]. Its main application is the clustering of available information. Its ability to preserve input data topology makes it a visualization tool that is widely used in various areas.
It can be represented as a two-layer structure: the input layer, whose function is only to allow information to enter the network, and the competitive layer, which is responsible for the clustering task. The neurons that form this second layer are connected and have the ability of identifying the number of "hops" or connections that separate them from each of the remaining neurons in this level.
Figure 1 shows the structure of a SOM network where the input layer is formed by a D-dimensional vector and the competitive layer has 9x7=63 neurons. Each neuron in this second layer has 8 direct neighbors (immediate connections). This connection pattern can change depending on the problem to solve. Each competitive neuron is associated to a weight vector

represented by the values of the arches that reach this neuron from the input layer. These values, for all the neurons in this layer, are represented in the figure by means of the W matrix.

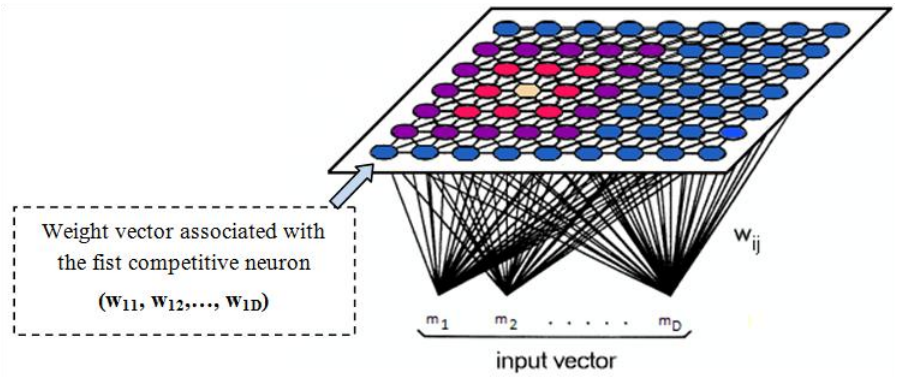

Weight vector associated with the fist competitive neuron
$(w_{11}, w_{12},\ldots, w_{1D})$
w_{ij}
m_1
m_2 . . . . .
m_D
input vector

Fig. 1. Classic structure of a SOM network

Network weights, W values, are initially random, but they adapt with the successive presentations of input vectors.
Since this is a competitive structure, each input vector is considered to be represented by (or associated with) the competitive neuron that has the most similar weight vector based on a given similarity measure.
The final value of W is obtained by means of an iterative process that is repeated until the weight vectors do not present any significant changes or, in other words, until each input vector is represented by the same competitive neuron than in the previous iteration.
In each iteration, the neuron representing each input vector is determined. This neuron is called "winning neuron", since it is the one that "wins" the competition to represent the vector (is the most similar one so far). Then, the weight vector for that neuron and its neighborhood are updated following equation (1).

$$w_{ij} = w_{ij} + \alpha * \left(x_i - w_{ij}\right) \qquad i = 1..n \tag{1}$$

where *j* is the competitive neuron whose vector is being updated and α is a value between 0 and 1 that represents a learning factor.
Equation (1) has variations that can be consulted in [15].
The concept of neighborhood is used to allow the network to adapt correctly. This implies that neighboring competitive neurons represent similar input patterns. For this reason, during the training process (obtaining W values) is started with a wide neighborhood that is then reduced as iterations occur.

Figure 2 shows the pseudo-code corresponding to the basic process for the adaptation of the SOM network.

```
W ← Random initial values.
Neighborhood ← set the size if the initial
neighborhood
NoIteReduction ← set the number of iterations that
must
                 occur to reduce the neighborhood
while termination criterion is not reached do
    for all each input vector do
           Input the vector to the network and
           calculate the winning neuron
           Update the winning neuron and its
           neighborhood
    end for
    Reduce the neighborhood {if applicable based on
    NoIteReduction}
end while
```

Fig. 2. Basic training pseudo-code for the SOM network

4. Proposed method

To be able to operate with the network described above, e-mails have to be represented by numerical vectors. To this end, a dictionary of terms will be built by processing an only text formed by the concatenation of the subject and body of each e-mail. Each word in the text is reduced to its root by applying a stemming algorithm [17]. This process is important for processing text in Spanish due to the syntactic changes related to gender, number, and tense. For example, words such as 'trabajo' (work), 'trabajar' (to work), 'trabaja' (he/she works), 'trabajos' (the works), 'trabajoso'(laborious) are reduced to the common root 'trabaj' by applying the stemming algorithm.
Once the root of each word is obtained, its frequency of use in the entire text and its average length are calculated. By means of statistical analysis processes, terms that are less relevant are discarded; the dictionary to represent e-mails is then built with the remaining terms.
Then, each e-mail is represented by a fixed-length binary vector. The number of elements in the vector is determined by the number of words in the dictionary. Each position will have a value of 1 if the word appears in the e-mail or a value of 0 if it does not.
Be D the number of words in the dictionary and M the number of available e-mails, each e-mail will be represented as follows:

$$mail_i = [m_{i1}, m_{i2}, ..., m_{iD}] \qquad i = 1..M \qquad (2)$$

$$m_{ij} = \begin{cases} 1 & if\ word_j\ is\ in\ e-mail\ i \\ 0 & otherwise \end{cases} \qquad j = 1..D \quad (3)$$

Using the vectors defined in (2), the SOM network is trained by applying the algorithm shown in Figure 2.

Be *N* the number of competitive neurons that form the SOM network, W_k will be the weight vector of the k^{th} competitive neuron. When the training stage is complete, the weights of these neurons will have the following format

$$W_k = [w_{k1}, w_{k2}, ..., w_{kD}] \qquad k = 1..N \qquad (4)$$

where

$$no.word_k = \{ s \in 1..M \,/\, \|W_k - mail_s\| < \|W_j - mail_s\|, \forall j, j \neq k \} \quad k = 1..N \qquad (5)$$

$$w_{kj} = \frac{\sum_{s \in no.word_k} m_{sj}}{\# no.word_k} \qquad j = 1..D \qquad (6)$$

Therefore, if the subset of e-mails represented by the same competitive neuron includes the same word, the vector that is associated with that neuron will also have a value of 1 in the position corresponding to that word.
In other words, the positions that have high values (close to 1) in the vector associated to a competitive neuron represent words that appear repeatedly in the emails that have this neuron as the winning one.
The method proposed in this paper uses the self-organizing maps to achieve two objectives: in the first place, to discard the words that are less significant, and secondly, to determine the most relevant word associations. Both these tasks are of interest, since the former helps not having to make an *a priori* decision regarding the size of the dictionary, and the latter is the solution to the problem presented.
The less significant words will be those words whose own weight is not enough to be clearly represented by a limited subset of neurons. This is the case of words that are combined with many terms or that are infrequently used. In either case, these are terms that provide little information, since in the first case they do not determine the subject matter and in the second case are not sufficiently supported (number of occurrences) to be considered significant. The trained SOM network is able to detect these words because they do not go beyond a minimum threshold in any of the vectors associated with the competitive neurons (Equation 7). Therefore, the vectors of W (Equation 4) are converted to binary values by using this threshold.

$$Wbin_k = [wbin_{k1}, wbin_{k2}, ..., wbin_{kD}] \qquad k = 1..N \qquad (7)$$

where

$$wbin_{kj} = \begin{cases} 1 & if\ w_{kj} > threshold \\ 0 & otherwise \end{cases} \qquad j = 1..D \qquad (8)$$

and irrelevant words are obtained with equation (9).

$$IrrelevantWords = \{word_j, j \in D \mid wbin_{kj} = 0, \forall k = 1..N\} \qquad (9)$$

It should be mentioned that each element of $Wbin_k$ will have a value of 1 at the positions corresponding to relevant terms. This allows identifying the frequent terms in the e-mails represented by the kth neuron of the network, which can be used to form various association rules.
An association rule is an expression with the following format

IF (antecedent) THEN (consequence)

where both the antecedent and the consequence are logical expressions referring to the words present in the e-mail.
The following are examples of association rules:

- IF ('board' ∧ 'exten' ∧ 'certific') THEN ('transcr' ∧ 'present' ∧ 'academ' ∧ 'approve')
- IF ('pending') THEN ('academic subject' ∧ 'certificate')

The first rule indicates that each time an e-mail has the words 'board', 'exten' and 'certific', the words 'transcr', 'present', 'academ' and 'approve' are also present. This rule refers to the approval by the academic board of extensions to present school transcripts. The second rule shows the relationship between the word 'pending' and the words 'academic subject' and 'certificate'. It also refers to pending academic subject certificates.
Rules are formed by combining in all possible ways the terms that appear in any given weight vector defined as in equation (7).

There are various metrics that can be used to determine the importance of a rule. The most common ones are:

- Support: It is the proportion of examples (e-mails) that fulfill the rule. For example, if the words ’pending’, ’academic subject’ and ’certificate’ are present in 300 e-mails from a total of 3,000 e-mails, the support of the rule

  IF (’pending’) THEN (’academic subject’ ∧ ’certificate’)

  will be 300/3000 = 0.1

- Confidence: it is the quotient of the number of examples that fulfill the rule and the number of those that only fulfill the antecedent. Let

  us consider again the rule IF (’pending’) THEN (’academic subject’ ∧

  ’certificate’) verified by 300 of the 3,000 available e-mails. Let us assume that after revising the available e-los, it is observed that 350 of those contain the word ’pending,’; the confidence of this rule will be 300/350 = 0.85

The importance of the rules obtained in this paper depends on the product of the two previously mentioned metrics. Therefore, the result that can be obtained is the interpretation of the most relevant rules.

5. Results

The method described in Section 4 was applied to the 2,995 e-mails from the Tutors Program (PACENI) between April and November 2009.
The initial dictionary was formed by 2,935 term roots; 287 of these were selected by statistical analysis. The selection criterion used had three stages:

i) First, words of atypical lengths were suppressed, considering as such all words whose value was more than 1.5 times the distance between the first and the third quartile (fourth dispersion). In the case of the PACENI e-mails, words that were longer than 18 characters or shorter than 3.5 characters were discarded. The average length was assessed based on all words that corresponded to the same root term.

ii) When analyzing the plot box corresponding to the occurrence frequency for each root term, it was observed that a large part of the population had a low value. That is, the most commonly used terms were the minority. Therefore, we decided to use those terms with extreme frequency. For the measured population, these were the terms with more than 49 occurrences.

iii) Finally, in the plot box of the reduced population, extreme values corresponding to the terms that are very frequently used in all e-mails are still observed. This reduces their importance. For this reason, those terms whose frequency was higher than 613—extreme value—were removed.

Figure 3 shows the plot box diagrams mentioned in ii) and iii).
A SOM network with 13x13 competitive neurons with 4 neighbors per neuron was used. The initial size of the neighborhood was set as a third of the number of rows in the network, that is, 4 neurons. This value is high, since it is the radius (number of "hop") that determines the area around the winning neuron where weight vectors are modified. The reduction was carried out every 30 iterations, with a maximum of 180 iterations. This value ensures that successive reductions will be carried out until the adaptation only affects the winning neuron.
After training the network, all weights that were not significant were removed from the matrix W; to this end, a threshold of 0.85 was used.
With the weight vectors of each competitive neuron, the combinations of terms that allow clustering the e-mails were determined.

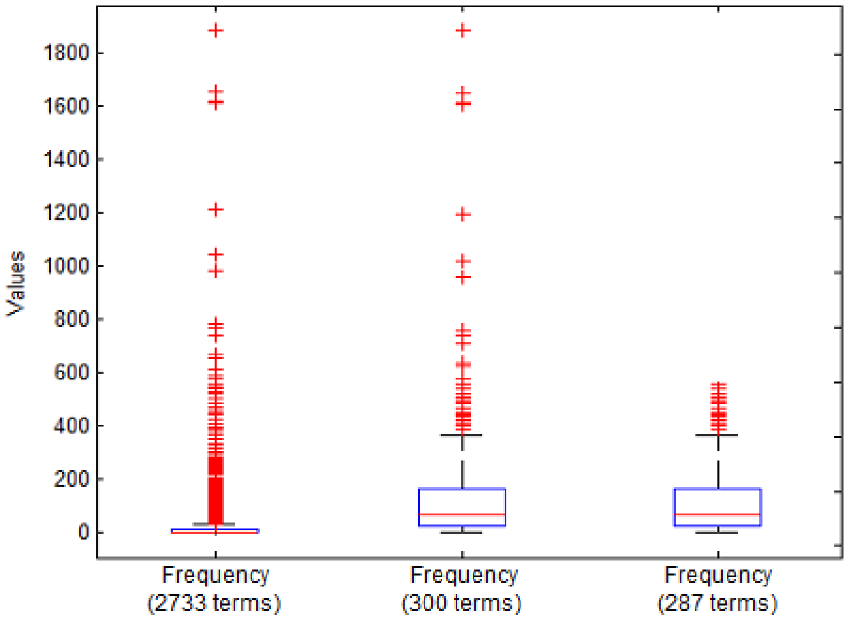


Fig. 3. Plot box diagrams corresponding to the successive reductions of dictionary terms as detailed in 5.ii) and 5.iii)

To measure their relevance, they were used to form the corresponding association rules, considering all possible combinations. The combination was associated with the maximum value obtained by multiplying its support and confidence values.
After 50 independent training sessions of the neural network, the most commonly occurring terms are the following:

('transcript', 'academic', 'approved', 'board', 'extend', 'present', 'certific')
('beta', 'classroom', 'inscription', 'English)'
('included', 'scholarship', 'ministry', 'ICTs', 'find', 'http', 'inscription')
('alumns', 'inscription', 'Guaraní', 'segundo')
('situation', 'know', 'tutor', 'question', 'contact')

These combinations appear in various orders, but always within the 20 first best positioned ones. This determines their importance within the set of e-mails. Another characteristic that was observed after the various tests is that the neural network allows discarding between 100 and 120 terms by means of the threshold function indicated in equation (8). This reduces considerably the time required to obtain the association rules to be measured.

6. Conclusions and future lines of work

An e-mail analysis mechanism based on data mining techniques has been presented. Even though the results obtained only refer to the 2009 Tutors Program of the PACENI, this analysis can be applied to other courses with no considerable changes.
Building the initial dictionary is essential to obtain good combinations of terms. The proposal presented in this paper included a statistical pre-processing so as to generate the dictionary as automatically as possible. This stage can be improved by manually entering additional information.
We are currently working with a dynamic SOM network so that adaptability is not limited. With this modification, we expect to solve the problem of neuron saturation. This is observed only in 0.2% of network neurons, but it may lead to the analysis of terms that are discarded with the current architecture.

References

1. Ananiadou, Sophia, Douglas B. Kell, Jun-ichi Tsujii (2006). Text mining and its potential applications in systems biology. Trends in Biotechnology. Elsevier Science London. Vol. 24, No 12, 571-579.
2. Krallinger, Martin, Alfonso Valencia (2005). Text-mining and information-retrieval services for molecular biology. Genome Biology. BioMed Central. Vol. 6, No 7, Article 224.

3. Ananiadou, Sophia, Paul Thompson, James Thomas, Tingting Mu, Sandy Oliver, Mark Rickinson, Yutaka Sasaki, Davy Weissenbacher, John McNaught (2010). Supporting the education evidence portal via text mining. Philosophical Transactions of The Royal Society A. 368(1925): 3829-3844.
4. Kuan C. Chen (2009). Text Mining e-Complaints Data From e-Auction Store With Implications For Internet Marketing Research. Journal of Business and Economics Research. The Clute Institute for Academic Research. Vol. 7, N. 5, 15-24.
5. Kin-Nam Lau, Kam-Hon Lee, Ying Ho (2005). Text Mining for the Hotel Industry. Cornell Hotel and Restaurant Administration Quarterly. Cornell Hospitality Quarterly. Vol. 46, No 3, 344-362.
6. Figuerola, Carlos G., Raquel Gómez, Angel F. Zazo Rodríguez, José Luis Alonso Berrocal (2002). Spanish Monolingual Track: The Impact of Stemming on Retrieval. Evaluation of Cross-Language Information Retrieval Systems. Springer Berlin/Heidelberg. Vol. 2406, 253-261.
7. Lorenzetti, Carlos M., Rocío L. Cecchini, Ana G. Maguitman, András A. Benczúr (2010). Métodos para la Selección y el Ajuste de Características en el Problema de la Detección de Spam. XII Workshop de Investigadores en Ciencias de la Computación, Área Agentes y Sistemas Inteligentes.
8. Mehrnoush Famil Saeedian, Hamid Beigy (2009). Dynamic classifier selection using clustering for spam detection. IEEE Symposium on Computational Intelligence and Data Mining.
9. Tsan-Ying Yu, Wei-Chih Hsu (2009). E-mail Spam Filtering Using Support Vector Machines with Selection of Kernel Function Parameters. Fourth International Conference on Innovative Computing, Information and Control.
10. Wanli Ma, Dat Tran, Dharmendra Sharma (2009). A Novel Spam Email Detection System Based on Negative Selection. Fourth International Conference on Computer Sciences and Convergence Information Technology.
11. Xiao Li, Junyong Luo, Meijuan Yin (2010). E-Mail Filtering Based on Analysis of Structural Features and Text Classification. Second International Workshop on Intelligent Systems and Applications.
12. Olivier de Vel (2000). Mining E-mail Authorship. In Proceedings of KDD 2000 Workshop on Text Mining.
13. Jason D. M. Rennie (2000). ifile: An Application of Machine Learning to E-Mail Filtering. In Proceedings of KDD 2000 Workshop on Text Mining.
14. Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, Anand Swaminathan (2006). Mining Email Social Networks. In Proceedings of ICSE 2006 Workshop on Mining Software Repositories.
15. Teuvo Kohonen (1997). Self-organizing Maps. 2nd Edition. Springer.
16. Teuvo Kohonen (1982). Self-organized formation of topologically correct feature maps. Biological Cybernetics. Springer Berlin/Heidelberg. Vol. 43, No 1, 59-69.

17. Barrenechea Pérez, Dennis D. (2006). A Spanish Stemming Algorithm Implementation in PROLOG and C#. Accessed at www.ai.uga.edu/mc/pronto/perez.pdf.

Prototype for the virtualization of group moderation based on the Metaplan technique

ALEJANDRO H. GONZÁLEZ[1], CRISTINA MADOZ[1], DAN HUGHES[2] AND MARÍA FLORENCIA SAADI[2]

Institute of Research in Computer Science. III- LIDI. School of Computer Sciences, Universidad Nacional de La Plata, Argentina.
[1]{agonzalez, cmadoz}@lidi.info.unlp.edu.ar.
[2] {florsaadi, danlaplata}@gmail.com.

Abstract. *In this paper, the research work carried out on group moderation strategies is presented, particularly the technique known as Metaplan technique. Through a specific visualization and questions technique, ideas and solutions for problems, opinion and agreement development, goal formulation, recommendations, and action plans are sought for. The development of this technique favors maintaining group motivation throughout the process. In this paper, the creation of a prototype is proposed for the virtualization of some stages with the purpose of expanding the training scope and facilitating team collaborative work. For the adaptation to the virtual format, the aspects of time, space, style, and rhythm of each student are considered, promoting their autonomy in the process. Finally, the initial results obtained with the prototype, which will be incorporated into a virtual teaching and learning environment, are presented.*

Keywords. *Metaplan, virtual environments, collaborative work, interaction.*

1. Introduction

Mankind has gone through various technology revolutions, namely, the agricultural and crafts revolution, the industrial revolution, the post-industrial revolution, and the information or knowledge revolution, which is currently ongoing.
The agricultural revolution was characterized by the use of animal strength, crop rotation, agriculture automation, and seed selection; the industrial revolution was characterized by the development of the textile and steel industries, the use of steam as energy, and the appearance of electricity; and the current revolution adopts as its basic development element information and communication technologies.

The Information and Communication Technologies (ICTs) applied to education are more than communication media and channels. They are modes to take possession of reality, the world, and knowledge. They allow generating virtual spaces where teaching and learning activities can be developed. When using ICTs, their technological characteristics and didactic possibilities should be taken into account. [1]
In the area of education, the presence of ICTs can be taken as a reality, and the possibilities, advantages, or results have become a matter of significance in the analysis by experts, the priorities of education administrations, and the changes suggested in the formation and ongoing training of teachers. [2] [3]
Group work becomes relevant with the incorporation of ICTs, and the interaction among several people can be used to achieve greater diversity in concepts and criteria [4]. People interacting with others usually are enriched with new opinions, and are able to draw new conclusions and tackle issues that would be more limited if the person worked alone or had only his or her own opinions or criteria. [5] [6]
The Metaplan technique can be considered as a group moderation methodology that provides, through visualization techniques and questions, a set of results in various fields of action such as planning, problem solving, participative decision making, needs diagnosis, group assessments, feedback, teaching and learning processes, debates, and workshops, among others. [7]

2. Collaborative Work

Collaborative work is defined as the set of intentional processes of a group to achieve specific goals, and the set of tools designed to support and facilitate such work [23]; for instance, in the context of an organization, group work with technological support is presented as a set of strategies aimed at maximizing the results and minimizing the loss of time and information to benefit organizational goals.
The greatest challenge is to achieve the motivation and active participation of the people involved in the process. Collaborative work or groupware are terms used to refer to the environment in which all project participants work, collaborate and help each other to carry out the project.
Belonging to a group with a common goal allows tightening ties among participants and generates in them a sense of belonging. Wikipedia is an example of collaborative platform whose purpose is the free dissemination of knowledge through the hard work of millions of users who update it on a daily basis.
From an information technology standpoint, the term “groupware” is defined by integrating the software and the human components. Groupware is not only a matter of technical issues; the organizational and social implications of introducing these new work tools should also be considered. It is more effective when software is adapted to support the goal of the group and the process used. The evolution of the human and the technological system must

be balanced so that the social implications of that progress are not forgotten, so that new organizational structures and roles are created [8].
From a pedagogical viewpoint, the central approach of Distributed Cognition can be mentioned, which tries to understand the organization of cognitive systems (people and environments). This theory wants to extend its limits beyond what is usually considered to be cognitive, and go into the realms of individuality to take in the phenomena emerging in social interactions as well as in interactions between persons and the structure of their environments. [9]
There are various elements that allow seeing the DCG. For instance, Internet may consist in making distributed cognition feasible, functional and relevant: a social network where users interact through publications and comments – a simple page with links can be a distributed cognition process. But maybe the most noteworthy aspect is how information technologies can technically make massive distributed cognition processes possible that would be very hard to organize analogically. [10]

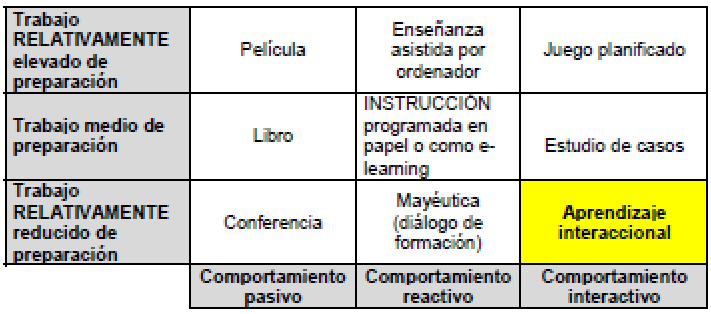
3. Metaplan

The Metaplan uses the maieutic method, of an inductive nature, based on dialectics (it assumes the idea that truth is hidden within the mind of each human being).
In essence, it consists in using dialogue to achieve knowledge. This method has various phases that start with a question asked to the other person and then the answer received is refuted through the use of general concepts, showing if there were mistakes or not in the reasoning process established, and arriving at a new concept, different from the previous one.
The basic idea of the Socratic method of teaching is that the teacher does not inculcate knowledge in the students, since their minds are not considered as an empty receptacle or box in which the various truths can be introduced; to Socrates, it is disciples who extract the knowledge from themselves [11]
The maieutic method offers interactivity – by requiring the exercise of one's own reason, people learn because they are actively involved in the process. [12]
Nowadays, it is considered that people can be strongly affected by their environment; this causes a change in the cognitive perspective, going from an approach centered on the individual processing of the information to a new approach that considers human agents and environments (including artifacts) while they are located in their contexts. Distributed Cognition (DCG) plays a special role in the understanding of the interactions between people and technologies. [13]
The Metaplan is an option for group work and is a method to lead meetings and group sessions of any type with the purpose of maximizing the interaction level of each and every participant.
It is developed through sessions that are coordinated by a moderator.
One of the basic principles presented in the Metaplan is the permanent visualization of the development of the meeting. Participants debate and take

down written notes of their ideas on cards that are placed on panels that everyone can see. Thus, in a short time, many contributions are obtained. The ideas contributed by others, always in sight on the panels, encourage the production of more ideas.
The moderator has techniques for each phase of the work, and offers the group a public strategy and various ways for posing questions and answering them, so that every participant can anonymously contribute to the evolution of the issues. The valuations, priorities, assessments, considerations, etc., carried out easily reflect opinion trends and generate transparency when going into decision making.
The easy access to work panels encourages participants to complete, modify and specify the concepts. The application of the Moderation Method carries a motivation effect: Since simple media are used (cards, labels, posters), it has some ludic characteristics that favor creativity and make meetings more enjoyable.[14]
In general, the method is used to manage large groups, but it can be adapted to small work groups. It can be used to create, collect, structure and visualize ideas. Priorities can be established and assessments can be carried out.
It can be used for people to present seminars to each other or to present various solutions to a case study.

3. 1 Description of the Technique

Metaplan is a set of "Communication Tools" to be used in groups that look for ideas and solutions for their problems, for the development of opinions and agreements, for the formulation of goals, recommendations and plans of action. It was conceived by Eberhand Schelle in Germany. The main pedagogical instrument is an interaction-type situation: based on a question or a thesis presented by the trainer, simultaneous responses, visible to all participants, are elicited. A tone of attention and tension can be maintained throughout the process due to the interest in checking if the other responses confirm one's own response, if they oppose it, or if they complement one's own knowledge on the topic being discussed.



Trabajo RELATIVAMENTE elevado de preparación	Película	Enseñanza asistida por ordenador	Juego planificado
Trabajo medio de preparación	Libro	INSTRUCCIÓN programada en papel o como e-learning	Estudio de casos
Trabajo RELATIVAMENTE reducido de preparación	Conferencia	Mayéutica (diálogo de formación)	**Aprendizaje interaccional**
	Comportamiento pasivo	**Comportamiento reactivo**	**Comportamiento interactivo**

Fig. 1. Relation between preparation work and behavior

The technique is considered to present a relatively reduced work for preparing the group and task, and it generates an interactive behavior among people. This can be seen in Figure 1
The trainer has to adopt the role of learning moderator or facilitator and is responsible for managing groups of people. One of the purposes to achieve with the Metaplan technique is the active participation of the students, by dividing complex problems in more limited problems and reducing the size of the groups. The entire group sets the task and then reviews the results. [15]
The Metaplan is divided in stages. In the first stage, the moderator decides the distribution of sub-groups and sub-topics among them.
In the following stage, the moderator collects the conclusions of each group and presents them to the entire group for all participants to work as one large group. When working with a group with several participants, several points of view can be generated. When analyzing all the ideas, they are grouped by similarity, generating "clouds of similar ideas".
When the process ends, a headline is placed on top of each cloud in order to distinguish them in the following discussions, and an order of importance is assigned to them. Thus, a map of the clouds is obtained, each cloud being formed by individual ideas that were grouped by mutual consensus.
Once the moderator collects the opinions of the participants, he/she groups them by similarity. For each new idea that has no similarity with the ones that have already been exposed, a new cloud is created; otherwise, it is grouped with the similar idea. Thus, the moderator puts together the clouds of ideas, generating a new sub-topic for each cloud that the moderator distributes among the participants.
In the next stage, each group puts together a “list of recommendations” (on-hold plans of action) – all these elements are included and the issues on which action should be taken are highlighted, in order of importance.
The entire group participates in the debate and a “list of actions” is generated in relation to the activities that can be carried out. Each action to be performed is assigned an owner and a group of people in charge of carrying it out.
Finally, the work is organized and the expected result is produced.

4. Virtualization of the Metaplan Technique

Nowadays, this technique is applied in various educational contexts. Even though it was originally created to be used in on-site classes, some trainers considered the possibility of using the methodology through a digital technology, where some of the stages are developed under a distance mode.
The technique was analyzed and the stages whose development through the Web is feasible were identified.
A review of the existing literature so far on the Internet did not yield any results for a virtual adaptation of the Metaplan that can be downloaded through the Web.

After analyzing the Metaplan, some aspects in which the Information and Communication Technologies (ICTs) can contribute to virtualization were identified:

- The possibility of expanding the scope of the training on the technique.
- Favoring the teaching and learning process in the development of the methodology, considering the aspects of learning time, space, style, and rhythm of each student, promoting their autonomy in the process.
- Taking advantage of the synchronous and asynchronous characteristics of ICTs to facilitate collaborative training events.

4.1 Proposed Development

This paper is part of the research work carried out for a graduate dissertation for the Bachelor's Degree in Computer Science of the UNLP. The development of an application that allows virtualizing the “discussion stages” and the subsequent presentation of the “list of recommendations” created by the sub-groups is proposed, so that people who cannot be present at all Metaplan sessions can be involved in courses that use this teaching methodology. Thus, part of the Metaplan is virtualized, allowing an assessment of the learning process through a record of the activities developed. [16]

The purpose is achieving an intuitive application that does not require the installation of additional software and has the necessary functionality to carry out the stages for creating a course, divide the participants in sub-groups, build the debate in each sub-group, incorporate the creation of the “clouds of ideas,” and design the list of recommendations.

The application allows creating integrated, collaborative activities through the Web [1].

The application presents an interface with various templates that adapt to the needs of the different users. There is an interface for the administrator that allows granting moderator rights to the users and configuring the sessions for each implementation of the Metaplan.

The moderator interface (Fig. 2) allows keeping track of the discussions and then building the list of recommendations.

The student interface allows students to work in collaboration by providing their contributions to build the future clouds of ideas.

There are two communication methods both for moderators and students – synchronous and asynchronous.

Both methods are private, that is, only the moderator and the participants from the specific sub-group have access to the discussions; groups cannot access other groups' discussions.

Asynchronically, the participants from a group can present a new opinion on a topic being discussed or associate a score (ranking) with an opinion presented by other participant of the sub-group. The discussions and subsequent conclusion are recorded in the system.

Synchronically, the participants can communicate among them and with the moderator, who may intercede in any event within the group discussion. Group members can consult with the moderator in case they have any doubts. To do this, a synchronous work plan of the "raising hand" type is designed.
As for templates, they allow configuring various communication possibilities within each work sub-group.

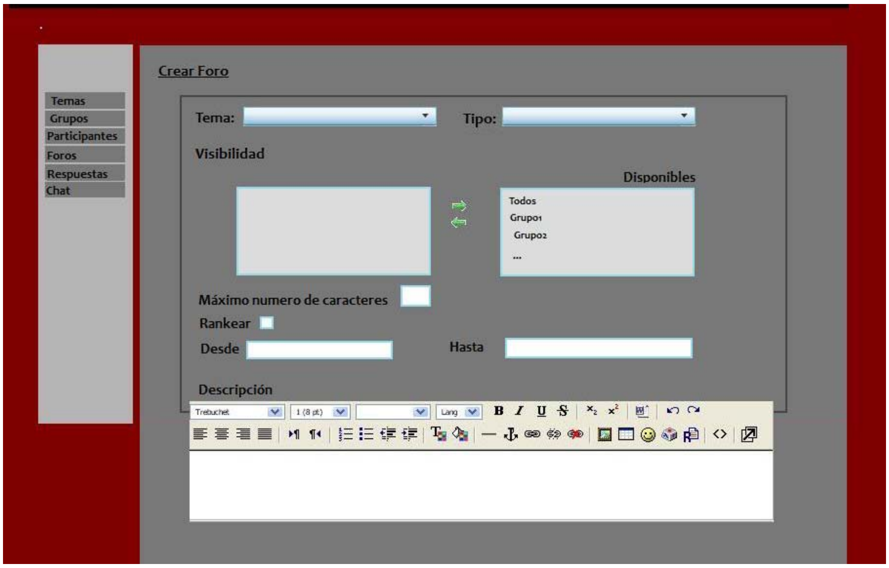


Fig. 2. Prototype for the moderator screen

4.2 Tools Used

After searching on the Web and then analyzing the tools for the implementation of the functionality described above, it was found that some of the tools that are suitable for this purpose are Php, MySQL and Symfony, which are described below.
Php (PHP Hypertext Pre-processor): it is a general-purpose, interpreted language that was originally designed for creating dynamic Web pages, and in particular for Web development, and it can be embedded within HTML code. It is generally run on a Web server, taking the PHP code as input and creating Web pages as output. Being an open software, it can be deployed in most Web servers and almost all operating systems and platforms free of cost. It is used to develop the code of the collaborative Web application and it is compatible with the tools to be used.
MySQL: It is a relational, multi-thread, multi-user database manager that is easy to use and fast. Since it is Open Source, it is one of the most widely used database engines on the Internet. It is used to manage user data and log the actions on the virtual stages of the Metaplan carried out by the different types of users.

Symfony: it is a full framework that is designed to optimize the development of Web applications through some of its main features. It separates business logics, server logics, and the presentation of the Web application. It provides several tools and classes aimed at reducing the time required to develop a complex Web application. Also, it automates the most common tasks, allowing the developer to fully focus on the specific aspects of each application.

5. Initial Progress

So far, the technique has been analyzed and described, the stages to virtualize have been isolated, information on other, possible virtualizations has been obtained from the Web, and contact has been established with the creators of the technique, who were very interested in the virtual development.
The interface is also being developed. Figure 3 shows a scheme of the elements that will be shown to participants on the screen.

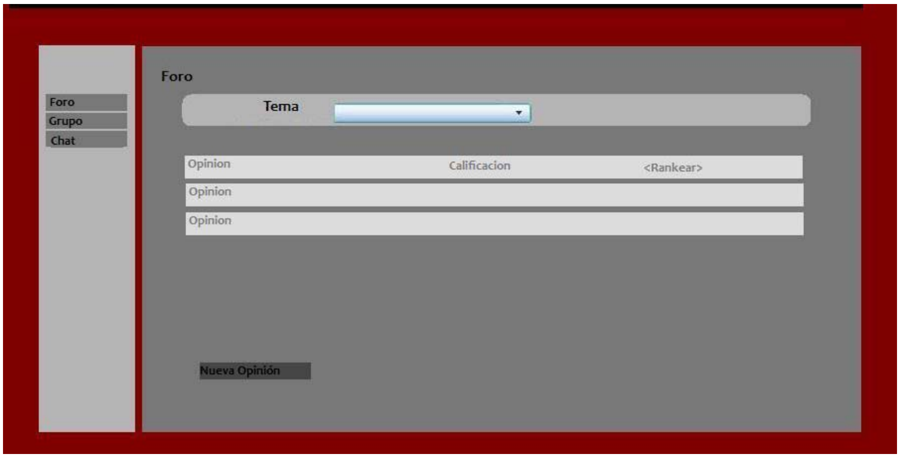


Fig. 3. Qualification and ranking scheme for participants

The tool is being built, and the necessary adaptations[1] for the forum and chat are being made in order to include the following tasks:

- **Single topic discussed by several groups:** this situation usually occurs when the cloud of ideas produces only one topic and the number of participants is enough to create several groups.
- **Several topics discussed by N groups**: this situation usually occurs when the cloud of ideas produces several topics and the number of participants is enough to create several groups.

[1] Source: Sigce An International Special Interest Group on Collaborative Editing, Publications, http://www.cit.gu.edu.au/~scz/sigce/.

Single topic discussed by a single group: this situation usually occurs when the cloud of ideas produces only one topic and the number of participants is not enough to create several groups.
Several topics discussed by a single group: this situation usually occurs when the cloud of ideas produces several topics and the number of participants is not enough to create several groups.

6. Conclusions and Future Work

Information and Communication Technologies (ICTs) are in the area of education to stay. Among the main concepts presented, that of group work can be mentioned, where people are enriched by the diversity of opinions and criteria.
The Metaplan technique and its future virtualization promote work group, and are intended to achieve an optimal use of Information and Communication Technologies to facilitate the teaching and learning process and promote time and space autonomy in this process.
The members of each sub-group of the Metaplan need a common workspace where they can develop their ideas, and this tool would allow them carrying out these activities remotely.
The technique is aimed at the assessment of the learning process, not just the end result. It provides virtual escort and a log of the actions carried out both by moderators (teachers) and students.
Some considerations regarding the continuation of research activities in this area:

- It would be desirable to virtualize a larger number of stages of the Metaplan technique so that students can manage their time and space with greater flexibility in relation to contents and technique phases
- Implementation of the virtual application of the Metaplan in various educational environments.
- Taking groups of students and assess the response and/or acceptance of the modifications to the technique introduced by the virtualization.
- Increase the abstraction level so that the application can be added as an educational module [4] or an activity in a CMS that meets SCORM standards.

References

1. Cabero, J. et al. (2000). Las nuevas tecnologías para la mejora educativa. Algunas comunicaciones y ponencias del Congreso Edutec99, Kronos, Seville.
2. Bates, A.W. (2000). Managing Technological Change, Strategies for Colleges and university leaders San Francisco, Ed. Jossey-Bass.

3. Roig, R.L, Marfil, A. (2002). Las Nuevas Tecnologías aplicadas a la educación. Elementos para una articulación didáctica de las Tecnologías de la Información y la Comunicación.
4. Moreno, F., Bailly-Bailliere, M. (2002). Diseño instructivo de la formación on-line, Ariel Educación, Barcelona.
5. Gregori, E. B., Badia A. (2005). Hacia el aula virtual: actividades de enseñanza y aprendizaje en la red, Revista Iberoamericana de Educación, Vol. 36, Nº 9.
6. Llorente Cejudo, M. del C., Cabero Almenara J. (2008). La Formación Semipresencial a Través de Redes Telemáticas (Blended Learning). Barcelona, Da Vinci, 243.
7. Cisnado Torres Xiomara. Metaplan, una metodología de diagnostico y moderación grupal. Centro de capacitación. Contraloría General de la República. Costa Rica. http://jaguar.cgr.go.cr/content/dav/jaguar/documentos/capacitacion/web_centro/Metaplan/metaplan.htm.
8. Rama, J. and Bishop, J. (2006). A survey and comparison of cscw groupware applications. Paper presented at the Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing couuntries, Somerset West, South Africa.
9. Solomon, G. (2005). "Distributed Cognitions. Psychological and educational considerations". Cambridge University Press. http://books.google.com.ar/books?id=m8Yna0cjxAgC&printsec=frontcover&source=gbs_summary_r&cad=0.
10. Madoz C., González A. Saadi M., Hughes D. (2010). Virtualización sobre un entorno de Enseñanza y Aprendizaje de métodos de trabajo colaborativo. Presentado en el TEyET 2010, Congreso de Tecnologia en Educación y Educación en Tecnologia. Calafate. Santa Cruz. Argentina.
11. Olleta J. Historia de la Filosofía. Volumen 1: Filosofía Griega. Editorial Edinumen. http://www.e-torredebabel.com/Historia-de-la-filosofia/ Filosofiagriega /Presocraticos/Mayeutica.htm
12. WIKIPEDIA, Mayeútica, retrieved on January 29, 2009 from http://es.wikipedia.org/wiki/May%C3%A9utica.
13. Hughes D., Saadi M., Madoz C., Gonzalez A. (2009). Aplicación para la administración y desarrollo de cursos con la técnica de Metaplan que aporta etapas virtuales mediante la Web. Presented in CACIC 2009. Congreso Argentino de Ciencias Informáticas y de la Computación. Jujuy. Argentina.
14. Hanusyk K.. Introducción al Método de Moderación.Vilassar de Mar. Barcelona, Spain. http://www.klaushanusyk.com/mod.htm.
15. Cisnado Torres Xiomara (2007). Virtualización de la Enseñanza-Aprendizaje de METAPLAN. www.infodesarrollo.ec/component/docman/doc_download/132-virtualizacion-de-la-ensenanza-de-aprendizaje-de-metaplan.html.
16. Lara, S. (2001). La evaluación formativa en la universidad a través de Internet, Eunsa, Barañáin.

VIII

Graphic Computation, Imagery and Visualization Workshop

DeLP Viewer: a Defeasible Logic Programming Visualization Tool

SEBASTIÁN ESCARZA, MARTÍN LARREA, SERGIO MARTIG AND SILVIA CASTRO

Laboratorio de Investigación y Desarrollo en Visualización
y Computación Gráfica, (VyGLab).
Departamento de Ciencias e Ingeniería de la Computación (DCIC),
Universidad Nacional del Sur (UNS),
Av. Alem 1253, Bahía Blanca, Argentina.
{se, mll, srm, smc}@cs.uns.edu.ar.

Abstract. *Defeasible Logic Programming (DeLP) is a knowledge representation formalism that combines results from Logic Programming and Defeasible Argumentation to provide reasoning based on contradictory and potentially incomplete information. DeLP allows information representation by using weak rules and provides an argumentation inference mechanism for warranting the entailed conclusions. It is necessary to comprehend the relationships between the arguments involved in DeLP derivation to understand the reasoning process and justify the replies provided by the DeLP system. In order to reach such a degree of understanding we present DeLP Viewer, a DeLP visualization tool for representing the inference process performed by a DeLP reasoner. Albeit there are many applications designed for argumentation visualization, our proposal provides a visual representation for the underlying logic and argumentative structure behind DeLP, allowing the interactive exploration of the entire reasoning process plus the internals of the involved arguments.*

Keywords. *Defeasible Argumentation Visualization, Defeasible Logic Programming, Dialectical Explanation, Hierarchy Visualization, Visualization.*

1. Introduction

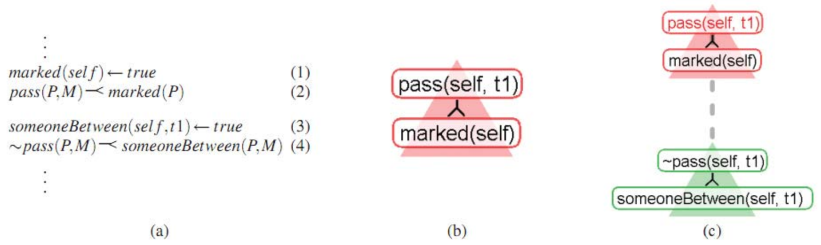
Formal modeling of real world problems has been object of study in Computer Science since its beginning. These models give to intelligent agents the ability to deal with information, reason about it, and infer new knowledge. However, when a real world problem is modeled, it is usual to have potentially contradictory and incomplete information.
DeLP is a knowledge representation formalism that combines results from Logic Programming and Defeasible Argumentation to provide reasoning based on potentially incomplete and contradictory information. It provides the possibility of representing information in the form of weak rules in a

declarative manner, and a defeasible argumentation inference mechanism for warranting the entailed conclusions [7].
The knowledge in a defeasible logic program can have intricate relationships that make complex the understanding of why some information is held and another is not. A set of dialectical trees representing the argumentation is built during the reasoning process. Such a set is called a dialectical explanation or δ-Explanation, and justify the status of a literal. The main worry for DeLP users is the construction of mental images about how arguments are confronted to derive conclusions, i.e., mental images about dialectical explanations.
In order to aid users to understand the justification of the derived conclusions, and to improve the analysis of the DeLP process, we present DeLP Viewer, a defeasible logic programming visualization tool. The goal is to assist users to gain insight into the dialectical process and, consequently, to easily design, understand, encode and debug defeasible programs. Our approach provides a node-link-based interactive visualization that preserves the internal rule-based structure of arguments, exploits the hierarchical structure of the dialectical tree, and allows on-demand exploration of the arguments content and their relationships.
This work is organized as follows. First we provide some preliminaries on DeLP to give the needed background to understand the problem we try to solve. Second we analyze related work in argumentation visualization and logic programming visualization. Third we present our tool, the rationale behind design decisions, the user domain constraints and the applied approaches. After that, we illustrate the main purpose of our tool in a concrete application scenario. Finally, we outline some conclusions.

2. Preliminaries on DeLP

DeLP is a logic programming language similar to *Prolog*, but it introduces additional constructors that enable the representation of potentially contradictory information and a defeasible argumentation inference mechanism. DeLP considers two kinds of program rules: *strict rules* used for representing strict (sound) knowledge, and *defeasible rules* used for representing defeasible knowledge, i.e., tentative information that may be used if nothing could be posed against it [7].
Rules in a DeLP program are combined to support or reject a claim. This combination is a logical derivation in which conclusions of some rules are used as premises of others using transitive logical dependencies. That combination of DeLP rules is an argument in favor of or against such claim (see Fig. 1). By contrast with other argumentation systems, arguments in DeLP are derived from the logic program and have internal structure. An argument is regarded as an explanation for a claim that is represented by a literal in such a logic program.
The claim of an argument may contradict some premise or the claim of another. In such a case, it is said that the contradictory argument attacks the

other. Each attacker can be further attacked by many other arguments and these relationships between arguments result in a tree-shaped structure called *dialectical tree* (see Fig. 1). Every argument attacked by at least one undefeated argument becomes *defeated*, and every argument without (undefeated) attackers becomes *undefeated*. If the root argument of the dialectical tree results undefeated, that dialectical tree represents an argumentation that supports the claim of such argument.



$marked(self) \leftarrow true$ (1)
$pass(P,M) \prec marked(P)$ (2)
$someoneBetween(self,t1) \leftarrow true$ (3)
$\sim pass(P,M) \prec someoneBetween(P,M)$ (4)
pass(self, t1)
marked(self)
~pass(self, t1)
someoneBetween(self, t1)
(a)
(b)
(c)

Fig. 1. Key concepts in DeLP. The figure shows a DeLP program fragment (a), an argument for *pass(self,t1)* that results from combining rules 1 and 2 (b), and a dialectical tree in which the root argument has been attacked and defeated (c). Rules in (a) have been numbered for referencing. Rules 1 and 3 are strict and rules 2 and 4 are defeasible. The literals labeled with *true* are removed from visualization for the sake of clarity.

Given a queried literal and a DeLP program, the DeLP reasoner builds a set of dialectical trees, trying to give support or contradict the arguments for or against the query. The sequence of obtained dialectical trees determines the reply of the system and is the explanation for such a reply. More detailed descriptions about the dialectical process behind DeLP can be found in [7] and [11].

3. Related Work

Although Nonmonotonic Reasoning, Logic Programming and Defeasible Argumentation are not novel fields of research, only a few years ago the DeLP formal definition was published [7]. Since then, an ever growing theoretical frame and some practical applications [6, 11] have been developed. However, previous specific work on visualizing DeLP does not exist. DeLP combines Logic Programming and Defeasible Argumentation. In both areas, there are many visualization approaches.
Logic Programming Visualization is a Software Visualization subfield. The main efforts in this area are devoted to Prolog language execution visualization and debugging. *AORTA* diagrams are widely accepted as graphical representation of Prolog execution [2, 5, 4]. However, these approaches have not direct applicability to our problem. DeLP users want to comprehend the dialectical process after the reasoner’s execution. They need

to understand logic dependencies and argument relationships at a higher abstraction level than those provided by *AORTA* diagrams. DeLP users do not want to visualize how logical variables are bound. They want to understand the argumentation for the system reply.
In Defeasible Argumentation, many visualization tools with different purposes have been developed [8]. The most relevant are Araucaria [9], Rationale [1], Avers [12], Belvedere [13], Athena [10] and Reason!able [14]. These argument visualization tools were thought to help in the visual building of argumentations by hand. The user adds arguments, establishes their support/rebut relationships, and no automatic processing is performed at this level. All these tools use node-link diagrams to represent the argumentation and the nodes are represented by boxes or circles. The arguments' content is plain (unstructured) text in natural language. Also, these tools share the presence of rebut and support concepts, but under different terminologies and red/green or red/blue color schemes are used to distinguish between them. Additionally, some of the applications mentioned above have semiautomatic and automatic layout algorithms to help users to keep arguments visually arranged. Finally, all these tools provide a basic subset of interactions containing argument selection, scrolling and zooming.
Nevertheless, these applications present drawbacks to be applied in DeLP, because they were designed to represent arguments in natural language. But in DeLP, the arguments have an underlying logic that must be represented to understand the dialectical process. Such a logic structure, internal to each argument, cannot be represented by ordinary argumentation visualization tools, and, in consequence, interactions that enable the exploration of such structures are not provided by them either.

4. DeLP System Overview

Before delving into the visualization design, we will outline the macro-structure and the typical working flow of the DeLP System. The DeLP system is composed by two main components: the DeLP reasoner and the DeLP Viewer. The former is responsible for deriving arguments from the defeasible logic program, building dialectical trees, and analyzing the defeating relationships in order to answer user queries. The latter is the visualization module and the tool presented in this article. Both modules are linked together by the dialectical explanation. This explanation is an XML file generated by the reasoner and received by our tool. Figure 2 shows an overview of the whole system.

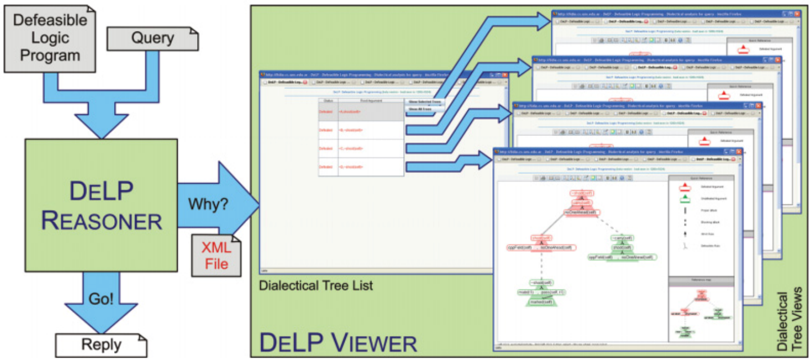


Fig. 2. DeLP System overview. The main components of the system and the typical working flow performed by the user are illustrated.

Given a defeasible logic program and a query, both specified by the user as plain text, the DeLP reasoner analyzes the logic program and derives a reply for the query. After that, the user can ask the system for examining the dialectical explanation by invoking the visualization module. Initially, the DeLP Viewer lists the dialectical trees involved in the reasoning process. The user selects the dialectical trees that will be visualized. For each selected tree, the system generates an interactive visual representation and shows it in a separate window. The details concerning each dialectical tree view are presented in the following sections.

5. Visualization Design

Visualization is concerned with two main aspects: building the visual representation and providing useful interactions [3]. We discuss the challenges and our solutions in the following sections.

5.1 The DeLP Viewer Visualization Pipeline

The backbone of our application is the DeLP Viewer visualization pipeline (see Fig. 3). For each dialectical tree to be shown, a new instance of this pipeline is configured and executed. The division of the process into stages results in a better design because the tasks throughout the pipeline can be addressed independently.

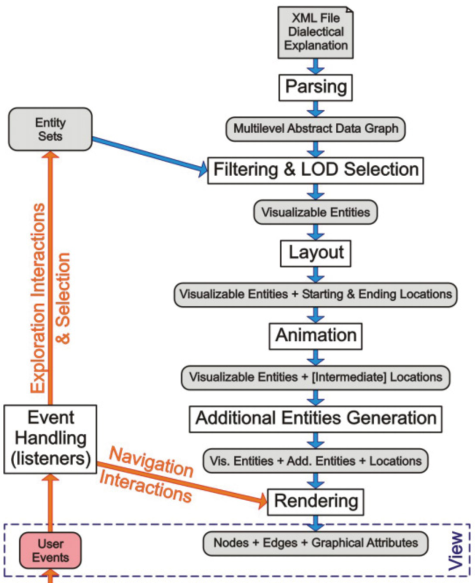

XML File Dialectical Explanation
Parsing
Multilevel Abstract Data Graph
Entity Sets
Filtering & LOD Selection
Visualizable Entities
Layout
Visualizable Entities + Starting & Ending Locations
Animation
Visualizable Entities + [Intermediate] Locations
Exploration Interactions & Selection
Event Handling (listeners)
Additional Entities Generation
Navigation Interactions
Vis. Entities + Add. Entities + Locations
Rendering
User Events
Nodes + Edges + Graphical Attributes
View

Fig. 3. The DeLP Viewer visualization pipeline. Many details were simplified due to illustration purposes. The figure shows the stages involved in data transformation from the explanation parsing to the view creation, and the modules implementing user interaction feedback.

The pipeline begins parsing the dialectical explanation XML file given by the reasoner. As the parsing result, a multilevel graph representation of the dialectical tree is built. The next stage in the pipeline filters entities that will not be visualized (e.g., the content of arguments that are not completely shown). After that, the layout algorithm defines starting and ending locations for each visible entity. Then, the animation stage interpolates these locations to get intermediate positions for the entities. After the animation stage, additional entities are generated, i.e., entities whose location is not calculated by the layout algorithm. Finally, in the rendering stage, the entities are drawn and graphical attributes like color are applied.
The user directly interacts with the tree-view. Some interactions are solved locally by only re-executing the rendering phase, and some others involve the manipulation of the entity sets establishing which arguments are completely shown, which ones are selected, etc.

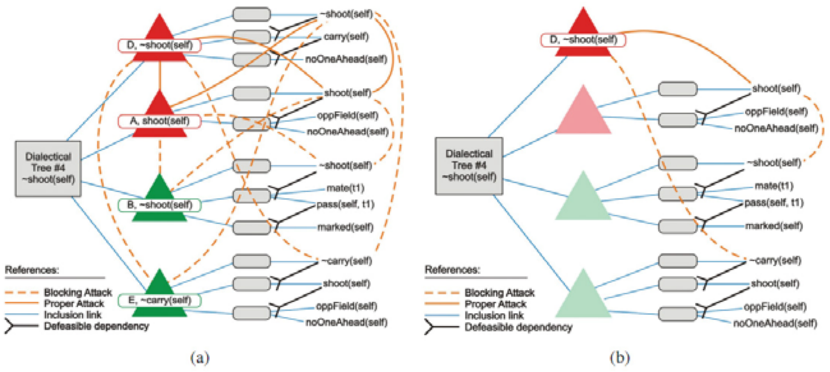


Fig. 4. The dialectical tree multilevel data structure with its four nesting levels. In (a) an instance of the underlying data structure for the dialectical tree shown in Fig. 6d can be seen. Apart from the inclusion relationships (drawn in blue), the edges for argument attacks and logic dependencies among rules are represented in this figure. In (b) only the visually mapped entities are shown. When an argument implosion is performed (as in the case of the argument *D* for *~shoot(self))*, its content loses their mapping.

5.2 The Dialectical Tree Multilevel Data Structure

One of the main benefits of our tool is the visual representation of the underlying logic structure. The data structure representing dialectical trees is a directed acyclic graph (DAG) that has four nesting levels. First, we have the whole dialectical tree which includes, in a second level, the arguments. In the third level there are nodes representing rule heads and bodies, and, in the last level, we have the literals. An instance of this structure for the dialectical tree shown in Fig. 6d can be seen in Fig. 4.
This structure represents a tree (i.e., the dialectical tree) whose nodes (i.e., arguments) are also trees. The arguments have a tree-like structure given by the underlying logic. Additionally, there is a third hierarchy: the inclusion tree itself. This tree constitutes a clustering scheme in which arguments can be thought as clusters that group rule heads and bodies.

5.3 The Visual Representation

The visual representation involves the definition of the spatial substrate, the visual elements used to represent arguments, relationships, logic dependencies, etc; and the graphical attributes of those elements. This implies the setting up of mappings from the abstract data to the graphical space. Additionally, our tool exploits well known representation conventions in DeLP to help users to feel familiarized faster with the visualization.
DeLP Viewer uses a node-link visual representation to show each dialectical tree. Literals, rule bodies and heads, and arguments are represented by nodes. Attack relationships and weak and strict logic dependencies are represented

by edges. Only nodes are positioned by our layout algorithm. Edges are straight lines that are placed using the nodes locations.
As was stated previously, a key aspect in our visualization is the underlying logical structure of the argumentation. This structure must be present in the visual representation to enable its reconstruction in the user mental map. Our approach places nodes in such a way that dependencies and relationships become evident and node overlap is avoided by means of a bounding box-based scheme. Additionally, the algorithm keeps the elements of each level aligned to enforce their position inside the hierarchy. These considerations facilitate the user insight. The DeLP Viewer visualization layout can be seen in Fig. 5.

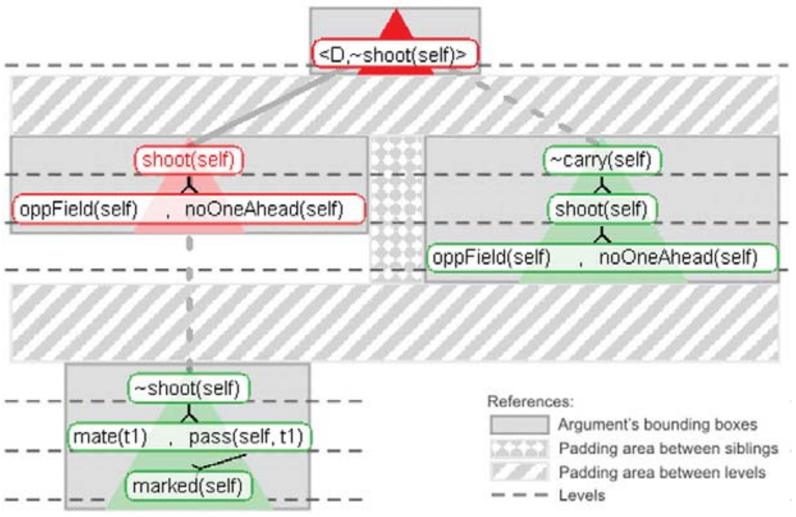

<D,~shoot(self)>
shoot(self)
oppField(self) , noOneAhead(self)
~carry(self)
shoot(self)
oppField(self) , noOneAhead(self)
~shoot(self)
mate(t1) , pass(self, t1)
marked(self)
References:
Argument's bounding boxes
Padding area between siblings
Padding area between levels
Levels

Fig. 5. The DeLP Viewer dialectical tree layout. This is the dialectical tree observed in Fig. 6d but here the root argument is imploded and, consequently, its content is not visible. Several additions were made to this screen capture for illustrating layout aspects.

Dialectical trees do not have any associated visual element because they are represented by more than one entity. Arguments are represented with triangles because the triangle enforces the idea of a tree-shaped hierarchy behind each argument. Rounded rectangles are used for representing rule heads and bodies and literal representation is performed through textual labels. DeLP Viewer shows defeated arguments in red and undefeated arguments in green according to the DeLP conventions.
Edges are represented by straight lines and rendered using the DeLP notation. To avoid confusion when many attacks are produced over the same argument, the attacked literals are represented with different color. Additionally, alpha blending is used in order to make exploded arguments

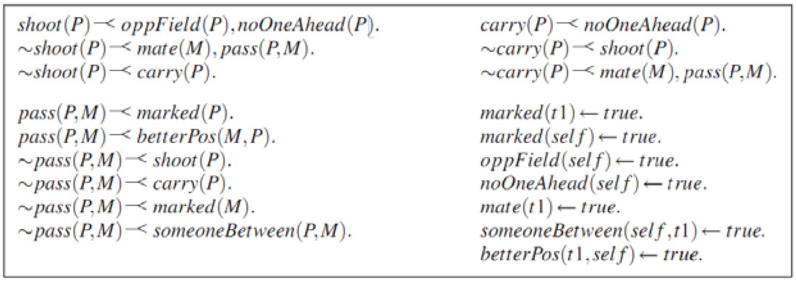
translucent enabling the visualization of the attacking edges under them. All these visual encodings can be appreciated in Fig. 6.

5.4 Interactions

Gaining insight about the reasoning process from a static representation is a difficult task. Interactions are needed to tune up the visualization and obtain richer perspectives that result in a better understanding of that process.
DeLP Viewer provides typical visualization interactions like *geometric zooming* and *scrolling*. An overview display is used to keep the user in context. Additionally, *argument selection* provides users with the ability to mark arguments to perform subsequent operations over them. Finally, DeLP Viewer provides two forms of semantic zooming. With *argument explosion/implosion*, the user can toggle between two argument representations: one shows only the literal supported by the argument and the other the entire argument content. Those different levels of detail are shown in Fig. 5 and Fig. 6d respectively. The other form of semantic zooming is *the literal label switching*. For each argument, the user can toggle between full length labels to get the whole information or abbreviated ones to reduce the visual complexity.

6. An Application Example

In this section we illustrate the practical usefulness of our proposal in a concrete application scenario. We consider a DeLP program representing the knowledge base (KB) for a robotic soccer player agent (more details can be found in [11]). The soccer agent uses DeLP in its decision making process.



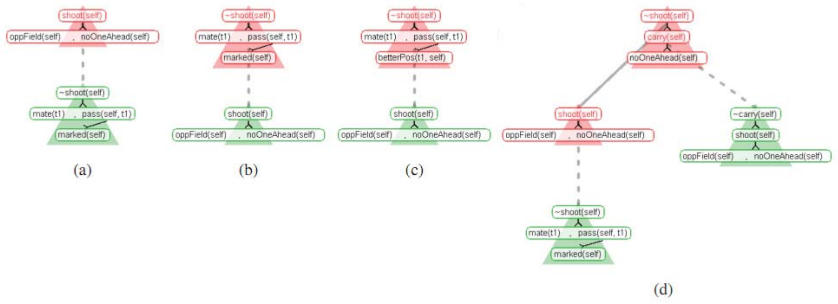
```
shoot(P) —< oppField(P), noOneAhead(P).
~shoot(P) —< mate(M), pass(P,M).
~shoot(P) —< carry(P).

pass(P,M) —< marked(P).
pass(P,M) —< betterPos(M,P).
~pass(P,M) —< shoot(P).
~pass(P,M) —< carry(P).
~pass(P,M) —< marked(M).
~pass(P,M) —< someoneBetween(P,M).

carry(P) —< noOneAhead(P).
~carry(P) —< shoot(P).
~carry(P) —< mate(M), pass(P,M).

marked(t1) ← true.
marked(self) ← true.
oppField(self) ← true.
noOneAhead(self) ← true.
mate(t1) ← true.
someoneBetween(self,t1) ← true.
betterPos(t1,self) ← true.
```

Prog. 1. DeLP program representing the KB of a robotic soccer player agent

Program 1 shows the KB of the soccer agent. In this case, defeasible rules were used to model the decision logic of the agent, and strict rules were used to represent the current situation of the agent, his teammate *t1*, and their opponents. The agent can find arguments for shoot, pass or carry the ball, and these arguments can engage in contradiction in ways that are not perceivable at first sight.

The main purpose of our tool is to show the justification for a given query. To illustrate this point, we will consider the query *shoot(self)* for the Program 1. From the examination of program rules it is difficult to venture an answer without a deeper analysis in which the dialectical explanation is rebuilt in some way. Must or must not the soccer agent shoot? If you ask to the DeLP System, it answers 'undecided'. The dialectical explanation for such a query and the Program 1 can be appreciated in Fig. 6.

After a quick examination of the dialectical explanation, the user can easily realize about how the argumentation proceeds, inducing dependencies and drawing conclusions. In the example, the reasons for and against shooting the ball are exposed. In Fig. 6a, a counter-argument for shooting the ball is found and it relies in the fact that our agent should pass the ball because he is being marked. In Fig. 6b and 6c the two arguments against shooting the ball are defeated because our agent is in the opposite field and there is no opponent ahead. Finally, in Fig. 6d, the argument against shooting that relies in carrying the ball is defeated by an argument against carrying the ball that suggests shooting instead. So, no argument can be built without at least one argument in contradiction, explaining, in that way, the reasoner's answer. Understand why this happens is not a trivial task from the program rules.



shoot(self)
oppField(self) , noOneAhead(self)
~shoot(self)
mate(t1) , pass(self, t1)
marked(self)
(a)
~shoot(self)
mate(t1) , pass(self, t1)
marked(self)
shoot(self)
oppField(self) , noOneAhead(self)
(b)
~shoot(self)
mate(t1) , pass(self, t1)
betterPos(t1, self)
shoot(self)
oppField(self) , noOneAhead(self)
(c)
~shoot(self)
carry(self)
noOneAhead(self)
shoot(self)
oppField(self) , noOneAhead(self)
~shoot(self)
mate(t1) , pass(self, t1)
marked(self)
~carry(self)
shoot(self)
oppField(self) , noOneAhead(self)
(d)

Fig. 6. A dialectical explanation. In (a) the only argument in favor of the query is defeated. So, the system analyzes the negation of the query. The three arguments against the query are also defeated (b), (c) and (d). Hence, the system answers 'undecided'.

Dialectical trees allow DeLP programmers to quickly analyze the foundations on which each argumentation line relies. The automatic building of such visual representations reduce the time and effort required in the dialectical explanation analysis, and enables the possibility of introducing modifications into the DeLP program and observe their effects in real time. In this sense, our tool has proven to be valuable in performing these application domain specific and typical tasks.

7. Conclusions and Future Work

We have presented DeLP Viewer: a visualization tool intended to help users to understand dialectical explanations. By contrast with other argumentation visualization tools, DeLP Viewer represents not only the dialectical argumentation, but also the underlying logical structure present in DeLP. This is a key aspect in order that users can gain insight about the relationships among arguments that justify the replies of the DeLP reasoner. The nested representation used is a novel aspect in Argumentation Visualization.
Major design decisions have been discussed along the paper. An important topic to emphasize is the use of DeLP drawing conventions to exploit the user's previous knowledge and improve the visualization experience.
As future work, we aim to include interactions intended to deal with some scalability issues of our tool, and to provide linking and brushing among dialectical trees to provide better visual queries. Additionally, we expect to perform some user evaluation of our tool. We need to obtain a quantitative measure of both the effectiveness of DeLP Viewer and the rate in which our tool improves the user experience.

Acknowledgement

This research was partially supported by the PGI 24/N028 and 24/ZN19, Secretaría General the Ciencias y Tecnología, Universidad Nacional del Sur, Bahía Blanca, Argentina.

References

1. Rationale by Austhink. http://rationale.austhink.com/
2. Brayshaw, M., M. Eisenstadt and J. Paine (1991). The Transparent Prolog Machine. Intellect Books.
3. Card, S. K., J. Mackinlay and B. Shneiderman (editors) (1999). Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann.
4. Domingue, J. (1998). Visualizing knowledge based systems. In J. Stasko, J. Domingue, M. Brown, and B. Price, editors, Software Visualization: Programming as a Multimedia Experience, chapter 16. MIT Press, January.
5. Eisenstadt, M. and M. Brayshaw (1998). The truth about prolog execution. In J. Stasko, J. Domingue, M. Brown, and B. Price, editors, Software Visualization: Programming as a Multimedia Experience, chapter 15. MIT Press.
6. García, A., C. Chesñevar, N. Rotstein and G. Simari (2007). An abstract presentation of dialectical explanations in defeasible argumentation. First international workshop on Argumentation and Non-Monotonic Reasoning (ArgNMR 07), 17-32, May.

7. García, A. and G. Simari (2004). Defeasible logic programming: An argumentative approach. Theory and Practice of Logic Programming (TPLP), Vol 4(1):95-138.
8. Kirschner, P. A., S. J. Buckingham Shum and C. S. Carr (editors) (2003). Visualizing argumentation: software tools for collaborative and educational sense-making. Springer-Verlag, London, UK.
9. Reed, C. A. and G. W. Rowe (2004). Araucaria: Software for argument analysis, diagramming and representation. International Journal on Artificial Intelligence Tools (IJAIT), Vol. 13(4):961-979.
10. Rolf, B. and C. Magnusson (2002). Developing the art of argumentation. a software approach. In Proceedings of the 5th International Conference on Argumentation.
11. Rotstein, N., A. García and G. Simari (2007). Reasoning from desires to intentions: A dialectical framework. Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-07), July.
12. van den Braak S. W., G. A. W. Vreeswijk and H. Prakken (2007). Avers: an argument visualization tool for representing stories about evidence. In ICAIL '07: Proceedings of the 11th International Conference on Artificial Intelligence and Law, 11-15. MIT Press.
13. Suthers, D., A. Weiner, J. Connelly and M. Paolucci (1995). Belvedere: Engaging students in critical discussion of science and public policy issues. In AI-Ed 95: Proceedings of the 7th World Conference on Artificial Intelligence in Education.
14. van Gelder, T. J. (2002). Argument mapping with reasonable. The American Philosophical Association Newsletter on Philosophy and Computers, Vol 2(1):85-90.

A Semantics-based Visualization Building Process

MARTÍN LARREA, SERGIO MARTIG AND SILVIA CASTRO

Laboratorio de Investigación y Desarrollo en Visualización
y Computación Gráfica, (VyGLab).
Departamento de Ciencias e Ingeniería de la Computación,
Universidad Nacional del Sur,
Av. Alem 1253, Bahía Blanca, Argentina.
{mll, srm, smc}@cs.uns.edu.ar.

Abstract. *A successful visualization allows the user to gain insight into the data in an effective way. Even with today's visualization systems that give the user considerable control over the visualization process, it can be difficult to produce an effective visualization. This paper is a step forward to achieve a visualization system that assists the user in the configuration and preparation of the visualization by considering both the semantic of the data and the semantic of the stages, through all the visualization process. In this article we present a system for the visualization of file system hierarchies where the color assignment and the configuration of the visualization technique are carried out by reasoning processes. This work set the way forward for the integration of reasoning in the visualization process.*

Keywords. *Semantic, Visualization, Ontology, Spherical Layout, Color Assignment, Reasoning, RDF, OWL.*

1. Introduction

Computer technology allows the visual exploration of large information resources ([1]). Huge amount of data is becoming available on networked information systems, ranging from unstructured and multimedia documents to structured data stored in databases. This is extremely useful and exciting; but the ever growing amount of available information generates cognitive overload and even anxiety, especially in novice or occasional users. Today, a wide range of users access, extract and display information that is distributed over several sources, which also differ in type, structure and content. In many cases, the user has an active control over the visualization process, but even then, it is difficult to achieve and effective visualization. A strategy to improve this situation is to guide the user in the selection of the different parameters involved in the visualization.
The Visualization field has matured substantially during the last decades; new techniques have appeared for different data types in many domains. With the use of visualization becoming more generalized, a formal understanding of the visualization process is needed ([2]). This work improves the one presented in [3] by including explicitly the semantic of the hardware, the user and the tasks in the visualization process.

Our contribution is a new step forward to achieve a visualization system that assists the user in the configuration and preparation of the visualization. Through a semantic reasoning we can determine all the parameters needed for the creation of a visualization. In our case we considered the visualization of a file system using the Spherical Layout ([4]).
The remainder of this paper is structured as follows. In the next section we give the foundation's details for our research. In Section 3 the previous work is detailed and Section 4 describes our semantics-based visualization model, including a brief description of the visualization application used to test it. In this section we considered the semantics of the data, the hardware, the user and the task. Finally, Section 5 summarizes the work providing some closing remarks and directions for future work. Because of space limitations we have not included an introduction to the Semantic Web and semantic reasoned terminology. For details about these concepts please see [3].

2. Semantics-based Visualization

Our main goal is the development of a visualization model that considers the semantics of both the data and the different stages in the visualization process. This model will transform data into information; according to Keller and Tergan ([5]), "information is data that has been given meaning through interpretation by way of relational connection and pragmatic context". This *meaning* can be useful. Information may be distinguished according to different categories concerning, for instance, its features, origin and relations. By making these considerations, the visualization process will be able to determine the characteristics of an effective visualization and guide the user through the different stages.
The user is an active participant in the visualization process and the goal of a visualization is to present data in a way that helps him to identify trends, features and patterns, generate hypotheses, and assign meaning to the visual information on the screen.
Since 2006 we have been working on the integration of semantic information into the visualization process ([6], [3]) and our main goal is to define an unified semantics for the data model and the process involved. In Section 4 we describe the semantics defined and the ontologies that represent them. In this section we also show how we created a visualization by using the results from the semantic reasoned and the ontologies.

3. Previous Work

There are some good examples ([7], [8], [9] and [10]) on how semantic information is integrated into the visualization tasks. However, in all these cases the role of the semantics is to improve the integration, querying and description of the visualization data; in neither case the semantics associated with the data is used to create the visualization or define its attributes. Only in [11] we can find a first approach to the use of the semantics as an aid to create a visualization. This work defines a customizable representation model which allows biologist to change the graphical semantics associated to the data semantic. The representation model is based on an XML implementation and used an XML Schema definition that prescribes its correctness and provides validation features. Unfortunately this work is only intended for biological use; it does not take advantage of the RDF or OWL representation and does not include any reasoning process with the semantic information.

4. Semantics-based Visualization Creation

A successful visualization allows the user to gain insight into the data. A successful visualization process takes advantage of the structure and the meaning of the data to create the most effective visualization. The structure of the data can be obtained from the data itself but not its meaning. Two sets may contain the same data, but if its meaning is different hen the final visualizations will not necessary be the same. This is why we included the semantic about the data, a way to describe the data about the data.

A visualization is greatly affected by what the user want to do with it. For the same data set, also with the same meaning, one visualization may be the most suitable for data exploration and another may be better for data comparison. By knowing what the user want to do and its meaning the visualization designer can create a better result. This is our reason to incorporate the semantic about the tasks.

Additionally, the response time of the interactions its crucial to obtain an effective visualization. If the user want to explore a 3D visual representation but there is no dedicated GPU on the computer, the user’s experience would be negatively affected. Besides that, a 4 inches screen cannot represent a visualization in the same way that a 42 inches screen does. A formal description of the system’s hardware could help the visualization designer to enhance the user experience with the visualization. Then, in addition to the data and task semantics, we also included the semantic of the hardware, a description of the actual system’s hardware.

All the previous semantics can be taken as input to the visualization process. All of them can change from one visualization process to another. But the visualization process can contain its own semantics as previous knowledge that depends on knowledge outside of the user scope. For instance, which colors combine better or which colormap to use to represent a data attribute. To demonstrate this we included the semantic of color.

Because of these considerations we extended our previous work ([3]) by improving our system’s architecture. Our previous work only included the color assignment process, but now we have also considered the rest of the visualization process, specifically the visualization technique configuration. We added new ontologies and included new steps where to apply the

reasoning process. As in our previous work, we used our Brows.AR application as test case.
In the next paragraphs we describe in detail these semantics and how we created the ontologies representing them. Then we detail how the reasoning process used these semantics in the visualization process. A review of our architecture can be seen in Fig. 1. We end this section with the description of how we adapted these elements to the Brows.AR application.

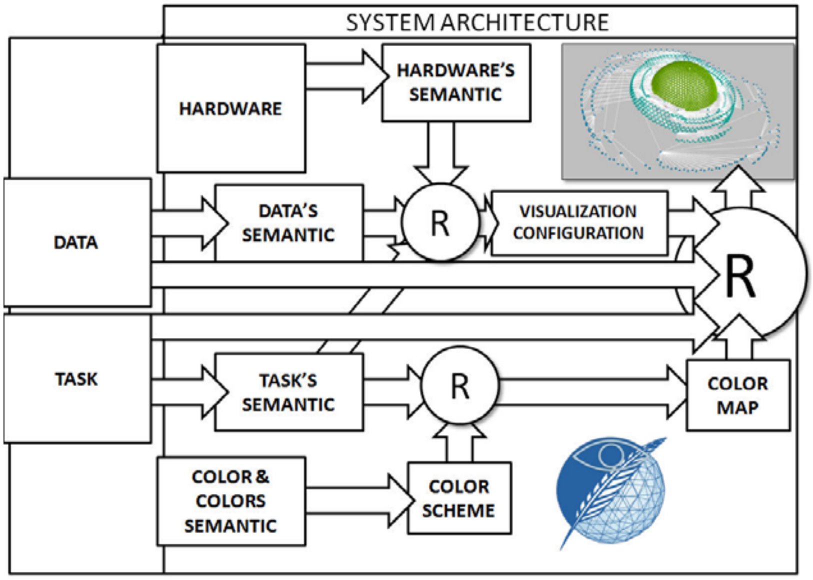


Fig. 1. The implemented system architecture

4.1 The semantic of the data

We created the semantic of the data based on metrics about the information to visualize. These metrics can give us information about the data itself. Because we used Brows.AR as a test case, a file system hierarchy visualization tool, our metrics are tree oriented. Our Data Ontology contains 5 metrics. All these metrics are data properties on a concept name *Metric.*

- Number of items *(n)*, in this test case the number of folders and files.
- Height of the tree *(h)*, number of items on the longest path from the root to the leaf.
- Width of the tree *(w),* maximum number of items on a level of the tree.
- Ratio of the tree, Height/Width, *(r).*
- Bounding box of the tree, Height*Width, *(bb).*

Because this is a test case, we limited the content of this semantic to include only metrics. The data's structure is implicit in the test case; a file system as a tree hierarchy. Each data element is a nominal one.

4.2 The semantic of the tasks

In our previous work ([3]) we showed how the color assignment could be accomplished by a reasoning process. In this paper, we extend that work to incorporate the semantic of the visualization tasks ([12]). For simplification we consider only one task, *filter*. As described in [12], *filter* is defined as: given some conditions on attributes values, select data cases satisfying those conditions. In our case, we use color to highlight those cases; particularly we change the color of the visual elements that are selected by the filter. Our goal was to describe, through an ontology, how to calculate the color property on each visual element.
By importing the previous ontologies, a developer can create its own Task Ontology using the concepts, relationships, properties and individuals related to these. As mentioned earlier, our task was filter with highlight using color. To stand out an object with color it is necessary to know which the background color is; and it is also important to know which color to use in the objects that will not be highlighted. To represent these elements, we included three concepts in the Task Ontology: *background, highlight* and *regular*. The *background* concept contains two object properties that relate to the *highlight* and *regular* concepts. *Background* represents the background color, *highlight* is the color for the filtered elements and finally *regular* is the color for the remaining elements. In order to set, in the ontology, that these last concepts are color we establish them as equivalent to the *Color* concept (See Section 4.4).
The benefit of this implementation is that the user is no longer responsible for the selection of the colors because a bad selection of the colormap may lead to a visualization where the highlighted elements do not seem highlighted because the contrast between the colors is not perceived. A novice user may choose colors based on what he thinks looks nice, but does not represent the true goal of the task.

4.3 The semantic of the hardware

As we said earlier, a visualization occurs in a context and in this case that context is the computer's hardware. The same visualization will not accomplish the same results if it is shown in a 4'' screen or in a 42'' screen. The effectiveness of a visualization method may depend on the available hardware and the peripheral devices attached to the computational system.
The complexity of the visual elements should be adjusted based on the 3D capability of the computer in order to improve the response time of the interactions. The Hardware Ontology contains a concept name *hardware* which contains the following data properties.

- An indicator of whether the computer has or has not a dedicated GPU *(GPU).*
- The height, on pixels, of the screen resolution *(hp).*
- The width, on pixels, of the screen resolution *(wp).*
- The number of pixels on display, *(pxs = hp*wp).*
- The size, on inches, of the computer display *(inches).*

4.4 The semantic of color

In this work, we expanded the work done in [3] to enhance the color representation. The new Color Ontology contains one concept, *color*. The role of this ontology is to express all the information related to color. The *color* concept contains 3 data properties, *red, green* and *blue.* Each one of these represents a primary color component. There are two object properties with domain and range in *color*, these are *next* and *opposite.* Based on the color wheel, it is possible to define, for each color, an opposite and neighbor. The opposite of a color *c* is another color *d*, whereas *d* is facing *c* on the color wheel. The next to a color *c* is a color *t* which is the following one to *c* on the color wheel, also known as neighbor. The concept *color* can be easily extended by new data and object properties.
We also created the Colors Ontology, a new ontology that imports the previous one. The role of the Color Ontology is to describe a generic color; the role of the Colors Ontology is to contain all the colors as individuals or instances of the *color* concept. All individuals contain specific values for their properties. For our test case we included 18 colors in the Colors Ontology.

4.5 The reasoning process

Having established the semantic elements in our architecture, we can now show how these elements are used by a reasoned to create results that will aid in the visualization creation. We began describing the role of the reasoning process in the color assignment process. The color assignment is accomplished using the Task Ontology through the reasoner. We then describe how the visualization technique is configure by the reasoner. In this stage the semantic of the hardware and data are used as input to the reasoning process. Finally we end with the description of the visualization creation *per se.*

Color Assignment Using Task Semantic. The Color and Colors Ontologies contain the formal representation of a color and all the colors as individuals. The first step for color assignment is to select a color scheme, to do so we ask the user to pick the visualization´s background color. Once this color is chosen, a color scheme is composed with the selected background color, its opposite and its neighbor, based on the Colors Ontology. There is no reasoned involved in this step.
With the color scheme and the semantic of the task, the reasoned can create a color map. The color map represents which colors will be used, and how.

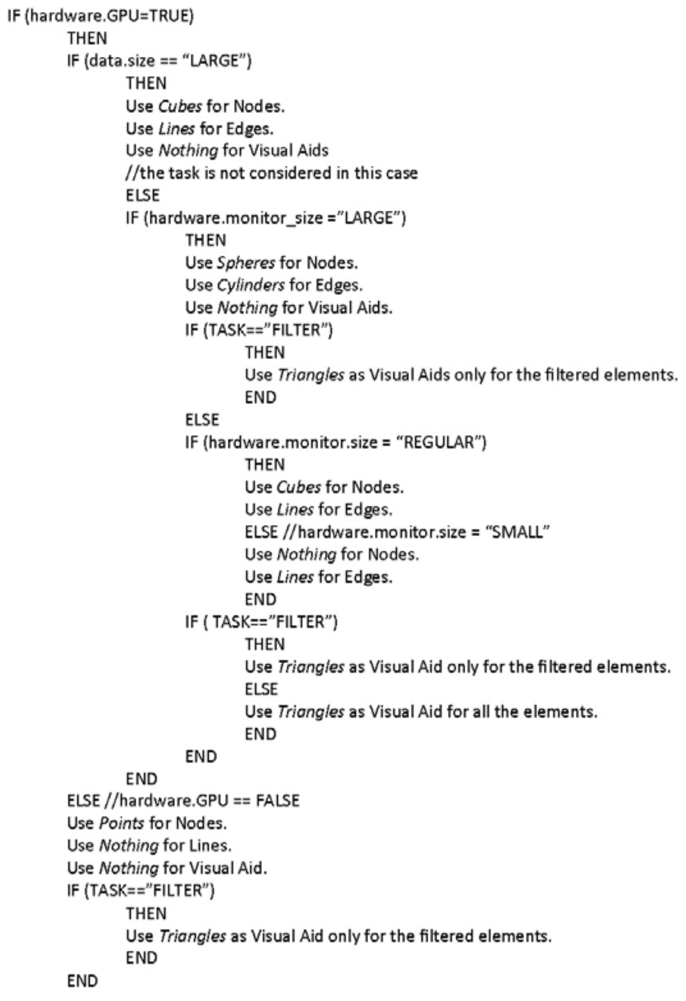
What the reasoner does is to take a color, the one selected by the user as background and to see that the concept *color* is equivalent to the concept *background* from the Task Ontology. Thus the reasoner knows that what holds true for a *color* and a *background* also does for the selected color, which is an instance of both concepts. The selected color has an *opposite* relationship and the reasoned knows that this is equivalent to the *highlight* concept in the Task Ontology.
Because of this, the color *opposite* to the selected one is the one that will be used to highlight elements in the visualization. The same process takes place for the *regular* relationship. The next step is to create the technique configuration based on the semantic of the data, the hardware and the task.

Visualization Technique Configuration using Semantics. The Spherical Layout technique supports different configurations of the final visualization. In our implementation of the layout there are multiples choices to graphically represent nodes, edges and visual aids:

- Nodes can be represented by a point in space, a cube or a sphere. The only visual property for points is color, so they are the less visual complex element in the technique. The sphere is the most complex one, follow by the cube; it is also possible not to map nodes visually. This gives us four possibilities of representation for the nodes: not mapped, points, cubes and spheres.
- Edges can be represented by two types of lines, a single line whose only visual property is color and a cylinder, which allows mapping more visual properties. The latest one is the most complex visual element. It is also possible not to map edges; For edges we have three possible representations: not mapped, lines and cylinders.
- In this implementation of the Spherical Layout the nodes are uniformly distributed on the spheres´ surface; to achieve this goal we discretized the surfaces of the spheres with triangles and place the nodes in the barycenter of some of these triangles. As a visual aid in the visualization, it is possible to show such triangles.

We defined a set of rules to relate the semantics of data, hardware and task with the configuration of the visualization technique. Based on the semantics defined earlier we created the rules shown in Fig. 2; in these rules we considered that the user may or may not want to perform the task *filter*. The result from the reasoning process in this stage is am instance of a concept called *configuration*; this concept indicates which visual elements to use in the visualization. This set of rules allows us to control how the visualization is created to make the most out of the current hardware.

Visualization Creation. The last intervention of the reasoned is in the creation of the visualization *per se*. The inputs to this process are the *configuration* concept, the color map and the data itself. As we did in [3], the reasoned can decide which color should be used for each data element, based on the *filter* task and the color map. Using the *configuration* concept the reasoner can determinate how the elements will be shown in the visualization. For this stage we created a new concept called *displayElement* which represents how a data element will be displayed. This concept includes data properties to handle the different choices for node and visual aid.



```
IF (hardware.GPU=TRUE)
          THEN
          IF (data.size == "LARGE")
                    THEN
                    Use *Cubes* for Nodes.
                    Use *Lines* for Edges.
                    Use *Nothing* for Visual Aids
                    //the task is not considered in this case
                    ELSE
                    IF (hardware.monitor_size ="LARGE")
                              THEN
                              Use *Spheres* for Nodes.
                              Use *Cylinders* for Edges.
                              Use *Nothing* for Visual Aids.
                              IF (TASK=="FILTER")
                                        THEN
                                        Use *Triangles* as Visual Aids only for the filtered elements.
                                        END
                              ELSE
                              IF (hardware.monitor.size = "REGULAR")
                                        THEN
                                        Use *Cubes* for Nodes.
                                        Use *Lines* for Edges.
                                        ELSE //hardware.monitor.size = "SMALL"
                                        Use *Nothing* for Nodes.
                                        Use *Lines* for Edges.
                                        END
                              IF ( TASK=="FILTER")
                                        THEN
                                        Use *Triangles* as Visual Aid only for the filtered elements.
                                        ELSE
                                        Use *Triangles* as Visual Aid for all the elements.
                                        END
                              END
                    END
          ELSE //hardware.GPU == FALSE
          Use *Points* for Nodes.
          Use *Nothing* for Lines.
          Use *Nothing* for Visual Aid.
          IF (TASK=="FILTER")
                    THEN
                    Use *Triangles* as Visual Aid only for the filtered elements.
                    END
          END
```

Fig. 2. Rules created to relate the semantics of the data, the hardware and the task with the configuration of the visualization technique

4.6 Brows.AR Application

We developed Brows.AR, an application for the visualization of file system hierarchies in 3D based on the Spherical Layout ([4]). The Spherical Layout is a 3D generalization of the Radial Layout. Instead of circles, as in Radial

Layout, we considered concentric spheres on whose surfaces we located the nodes. In the Radial Layout each node, except the root, is allocated in a 2D sector within the sector assigned to its parent; in the Spherical Layout we considered a spherical wedge and the nodes are allocated on the surfaces defined by this wedge.
In order to test our layout, we created a 3D representation of a file system structure; to enrich the visual representation we allow the user to see the triangles that were used to place the nodes. These triangles are painted with the same color used for the nodes but with a high level of transparency. The color of each node is based on the file type that the node represents. In the case of very large trees, it is possible to remove the nodes and edges from the visual representations and to leave only the triangles, providing an overview of the hierarchical structure and improving the application performance. For details about the implementation and the supported interactions see [4].

4.7 Brows.AR Semantic add-on

In order to integrate the semantic information with our application we created a class called *Reasoner*; its main method was *ask*. The *Reasoner* class used Protégé ([13]) and Jena ([14]) APIs to interact with the ontologies. The reasoning service was provided by the Pellet ([15]) API. The constructor of the *Reasoner* class takes one parameter, a *JenaOwlModel* which is a representation of an ontology model. To improve the performance of the last stage in the visualization process, we used a hash table as a cache memory to keep the information retrieved from the reasoned. If a particular data element is not in the cache, the application asks the reasoned for the corresponding d*isplayElement* instance; then the pair *(data element, displayElement instance)* is saved in the cache. Because all the edges are handled uniformly, the reasoned asks only once about this option at the beginning of the process.

5. Conclusions

We designed several ontologies related to the semantic of the visualization process. We included the semantic of the data, the tasks, the hardware and the color. Within the visualization process, we used a semantic reasoned to create the final visualization. This architecture was integrated into the Brows.AR application, a 3D visualization of file systems. With this integration, we created a visualization system that was able to assist the user in the preparation and configuration of the visual representation.
A visualization system should ensure that, even if the user is not a visualization expert, the generated visualization will be the most suitable for the user and the data domain. This work presents a break trough in the visualization research because of the integration between the visualization process and the knowledge on visualization creation. The used of Brows.AR

as a test case proved that it is possible to use a semantic reasoner to create a visual representation.
As future work, we are looking forward to include Strahler number and its bifurcation ratio as part of the semantic of the data. The semantic of the hardware will be extended to include more input and output capabilities and we will expand the semantic of the task to include all the tasks described in [12].

Acknowledgement

This research was partially supported by the PGI 24/N028 and 24/ZN19, Secretaría General the Ciencias y Tecnología, Universidad Nacional del Sur, Bahía Blanca, Argentina.

References

1 Edward, R. (1986). The Visual Display of Quantitative Information. Graphics Press, Cheshire, CT, USA.
2 Duke, D. J., Brodlie, K. W., Duce, D. A. (2004). Building an Ontology of Visualization. In Proceedings of the conference on Visualization '04 (VIS '04). IEEE Computer Society, Washington, DC, USA.
3 Larrea, M., Martig, S., Castro, S. (2010). Semantics-based Color Assignment in Visualization. In: Journal of Computer Science and Technology, 14-18, ISTEC.
4 Larrea, M., Martig, S., Castro, S. (2007). Spherical Layout – Layout for 3D Tree Visualization. In: Proceedings of the IADIS 2007, Multi Conference on Computer Science and Information Systems, 91-98.
5 Keller, T., Tergan, S. (2005). Visualizing Knowledge and Information: An Introduction. In: Lecture Notes in Computer Science, vol. 3426/2005, 1-23.
6 Larrea, M., Martig, S., Castro, S. (2010). Visualización Basada en Semántica. In: Proceedings of the XII Workshop de Investigadores de Ciencias de la Computación. 270-274, Red UNCI.
7 Zhao Xu, Huajun Chen, Zhaohui Wu (2005). Applying Semantic Web Technologies for Geodata Integration and Visualization. In: Proceedings of ER (Workshops). 320-329.
8 Kashyap, V., Cheung, K.-H., Doherty, D., Samwald, M., Marshall, M., Luciano, J., Stephens, S., Herman, I., Hookway, R. (2008). An Ontology-based Approach for Data Integration - An Application in Biomedical Research. In: Real-world Applications of Semantic Web Technology and Ontologies, Springer-Verlag, Heidelberg.
9 Kalogerakis, E., Christodoulakis, S., Moumoutzis, N. (2006). Coupling Ontologies with Graphics Content for Knowledge Driven Visualization.

In: Proceedings of the IEEE conference on Virtual Reality (VR '06). IEEE Computer Society, Washington, DC, USA, 43-50.
10 Weng, Z., M., Bell, D. (1998). Integrating Visual Ontologies and Wavelets for Image Content Retrieval. In: Proceedings of the 9th International Workshop on Database and Expert Systems Applications (DEXA '98). IEEE Computer Society, Washington, DC, USA, 379.
11 Bouzeghoub, M., Goble, C., Kashyap, V., Spaccapietra, S. (eds.) (2004). Semantics of a Networked World. Semantics for Grid Databases. First International IFIP Conference on Semantics of a Networked World: ICSNW 2004, Paris, France, June 17-19.
12 Bongshin, L., Plaisant, C., Sims Parr, C., Fekete, J., Henry, N. (2006). Task taxonomy for graph visualization. In: Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization (BELIV '06). ACM, New York, NY, USA, 1-5.
14 Protégé ontology editor and knowledge-base framework. http://protege.stanford.edu/
15 Jena – A Semantic Web Framework for Java. http://jena.sourceforge.net/
16 Pellet: OWL 2 Reasoner for Java. http://clarkparsia.com/pellet/

VII

Software Engineering Workshop

Using DEVS for Evaluating Software Architectures

VERÓNICA BOGADO, SILVIO GONNET, HORACIO LEONE

INGAR - CIDISI, Facultad Regional Santa Fe, Universidad Tecnológica Nacional,
CONICET.
Avellaneda 3657, 3000 Santa Fe, Argentina.
{vbogado, sgonnet, hleone}@santafe-conicet.gov.ar.

Abstract. *In this work a model for the simulation of software products at early stage of the software development process is proposed, using its software architecture. It is focused on the representation of required architectural concepts and its translation into simulation model components. Thus, we proposed the use of DEVS formalism with the purpose of adding simulation advantages to the context of the architectures design. Unlike other simulation techniques, DEVS framework allows to keep the model uncoupled from the simulator and a modular and hierarchical form of building. The proposal describes how architectural elements are transformed into simulation elements. The main goal is to acquire quantitative information for evaluating the quality attributes of the system at the design stage, so decisions can be taken earlier in the development process.*

Keywords. *software architecture, evaluation of architectures, DEVS.*

1. Introduction

Today, it is known that intermediate products obtained at the first stages of the software development, such as requirements specification and software architecture, are very important for the final product. Software architecture design provides foundations for analyzing information related to software quality. However, to carry out this analysis with a trade-off between quality attributes requires architects that have knowledge of different techniques. Thus, companies have difficulty to find human resources with the required "know-how". In this context, the development of methods and tools that provide a support to architects at the software architecture design has an increasing importance.

The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them [1]. It is a complex entity and cannot be described in one-dimension, so for a good analysis it is essential different perspectives or views, like proposed by Clements [2], Hofmeister [3] and Kruchten [4]. Although there is no consensus, these authors agree that it is important to model at least a static view and a dynamic view. Furthermore, several alternatives were proposed to represent these views. For example, Use Case Maps (UCM) can be used to represent functional aspects in software

architectures [5]. This notation adds responsibilities to the architectural structures, where these responsibilities have causal relationships, representing behavior and structures at a high level of abstraction [6].
The different representations of the architecture allow evaluating quality attributes related to the software. Several works for analyzing software architectures with different purposes have been proposed. Some of them are focused on the architecture, analytical form based on the stakeholders and the expert; others are focused on quality attributes, proposing a quantitative or qualitative analysis based on Markov Decision Process, Petri Net or Queuing Nets ([7], [8], [9]), and others are focused on scenarios. In the last group, there are works based on patterns (analysis of the architecture done by the expert), others based on design decisions (documentation and analysis of early decisions that affect the evaluation of quality attributes) and others based on Use Case Maps (UCM) employing Markov Process or Queuing Theory [10].
However, few tools to analysis the dynamic of the system at early stages, showing how changes could happen, have been proposed. Simulation is a powerful tool for analyzing the states system could pass and for obtaining values to validate different scenarios. The variables that characterize an entity, i.e. values that allow to measure performance, can be modified and studied, showing the impact of a change without implementing the system.
In this way, a model for simulating software products at early stage of the software development is proposed, using its software architecture. So, the behavior of the system can be studied, calculating measures that are used to validate quality attributes specified in the software requirements.
In this paper, we propose a representation of information related to software architectures that is required to a complete study of software quality. We also consider the description of the corresponding simulation elements. In this way, we propose the use of Discrete Event System Specification (DEVS) [11], formalism developed by Bernard Zeigler and it is widely used as an object-oriented modeling and simulation tool. DEVS represents models in a modular and hierarchical form, providing a powerful ability of expression and abstraction. These characteristics suitably represent concepts of software architecture and their relationships.

2. Conceptual Model: Software Architecture for Evaluating Quality Attributes

The proposed model represents (as shown in Fig. 1) information for the analysis of the architecture from a dynamic viewpoint including data for calculating metrics about different quality aspects. Thus, the behavior of the system related to the structural aspects of the architecture can be understood, analyzing interactions between components that have some presence at runtime. This model represents essential elements, following different approaches in the literature ([1], [2], [3], [12]), and functional aspects, using UCM [6]. This is the basis for the simulation.

In a dynamic view different architectural elements are represented. *ArchitecturalElement* (Fig. 1) is an abstract concept, which represents structures that have presence at runtime. Two important concepts are specialized from it, component (*Component* concept in Fig. 1) and connection mechanism (*ConnectionMechanism* concept in Fig. 1), which is the communication between two or more components. If connection mechanisms are complex connectors, they can have assigned responsibilities to them.

The proposed model includes *SimpleComponent* concept (Fig. 1) for representing simple structures. Simple component is a software entity that could have some run-time presence (such as process, object), and it is in charge of a set of responsibilities. A more complex structure is represented by C*ompositeComponent* (Fig. 1), which depends on a set of components for carrying out its responsibilities. It can be composed by both simple components and composite components, delegating the responsibilities to them, because responsibility concept is only represented in the atomic level (*SimpleComponent*).

Components, connection mechanisms and responsibilities concepts represent essential elements to build software architectures, describing the system behavior too.

A responsibility (*Responsibility* concept in Fig. 1) is a statement about software objects. It could be an action an object performs, knowledge an object maintains, or a major decision an object makes that affect others [13]. The responsibilities have *cause-effect* relationships (*causes* relationship in Fig. 1), where the fulfillment of one or more responsibilities implicates the execution of others [6].

Architects specify values related to metrics required to evaluate quality attributes. So, quantitative aspects are associated to the responsibilities, smallest units at runtime. Quality attribute value (*QualityAttributeValue* concept in Fig. 1) depends on the measures that are used to analysis quality scenarios; it is the result of a simple or complex metric. *Measure* concept (Fig. 1) maintains information about the needed values for calculate a quality indicator.

Finally, aspects related to the decisions of the architect are captured in the model as *Pattern* concept (Fig. 1) and *Tactic* concept (Fig. 1). A *Pattern* collects a set of architectural elements, structures, to conform a unit in itself with characteristics that respond to decisions of the architect. A pattern packages a set of tactics. A *Tactic* is a design decision that influences in the control of the quality attribute response, in other word, it is a design option.

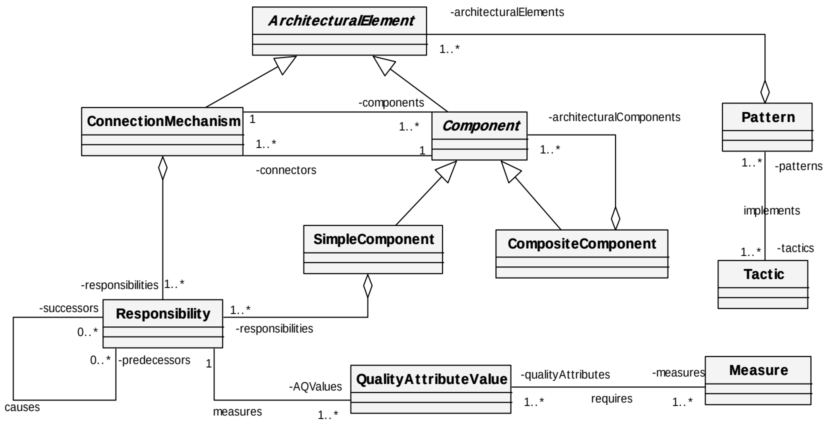


Fig. 1. Conceptual Model of the Software Architecture Domain

3. Hierarchy of DEVS for Architectural Elements

The rules for transforming the main architectural elements into simulation elements are summarized in Table 1. This work is focused on the specification of DEVS models for representing the basic structures of the software architecture domain; especially we describe simple component and responsibility concepts, because they are the most used in an architectural description.

Table 1. Relationship between models. Correspondence between conceptual elements (software architecture) and simulation elements (Hierarchy of DEVS-architecture domain)

Conceptual Model Elements	Simulation Model Elements
Responsibility	Atomic DEVS with ports
SimpleComponent	Coupled DEVS
ConnectionMechanism	Coupled DEVS
Relation: *causes* (between responsibilities)	I/O Ports, Couplings between ports
Relation: aggregation (*SimpleComponent-Responsibility*, *ConnectionMechanism-Responsibility*)	Couplings between ports

Following guidelines suggested by Zeigler [11] and other authors [14], a hierarchy of DEVS models is built. The elements or building blocks are defined according to some features: self-contained (local information and process), interoperable (cooperation between blocks), reusable (multiples instances), replaceable (interchangeable in the model), and encapsulated (internal structure and interfaces).

3.1 DEVS Model for Responsibility Concept

Responsibility concept (Fig. 1) is transformed into a responsibility model (*RM*), an atomic DEVS with ports (as shown in Table 1). The relationships between responsibilities (*RM*) in the simulation model are given by the input/output ports.

Each *RM* calculates its output values, some related to its own states and others related to the measurement of quality attributes. At present, parameters included in the simulation elements allow to measure performance aspects and validate different performance scenarios. Each element returns the execution time for computing the turnaround time of the system when stimuli arrive.

$$RM = \left(X_{RM}, Y_{RM}, S_{RM}, \delta_{RM,ext}, \delta_{RM,\mathrm{int}}, \lambda_{RM}, ta_{RM}\right). \qquad \textbf{(1)}$$

3.1.1 Set of Input Events (X)

The set of inputs represents information about fulfilled responsibilities and indicates the level in which each predecessor responsibility is completed. The only required information for activating a responsibility initially is to know if previous responsibilities have finished their executions. This set could be increased by adding more details to the model to evaluate other quality aspects of the system.

Set of input ports:

$RIP = \{prip\}$ where *prip* is an input port that is connected to the output ports of other responsibilities or to input ports of coupled models (being a component).

Set of input values for *prip*:

$X_{\mathrm{RM,prip}} = \{finished\}$

Set of input ports and input values pairs:

$X_{RM} = \{(prip, finished)\}$ where $prip \in RIP$

3.1.2 Set of States (S)

A responsibility reflexes a point where the system makes a change in its state, because requests or actions from other elements affect it. So, the states provide information to the designer for analyzing if it is activated, executing or in a passive mode.

Phases:

- *inactive*: passive state, waiting for an external event. The system stays in this phase until an event occurs and interrupts the system condition.
- *active*: transitory state. This phase starts an internal transition that generates an output. It indicates that the execution of the responsibility

has been started. The duration of this phase is null and it cannot be interrupted by external events.

- *executing*: this stage indicates the responsibility is being performed, where execution means the processing of code in software domain.

σ: resting time in a given state.

$$S_{RM} = \{inactive, active, executing\} \text{ x } R_0^+ . \quad \textbf{(2)}$$

Parameter (fix parameter of the model): *execution_time,* time that an architectural element needs to carry out a responsibility. It follows a probabilistic distribution.

3.1.3 Set of Output Events (Y)

A responsibility is in charge of emitting two types of information. One related to its state, information of interest for the successor responsibilities, and other related to the values used for measuring quality attributes.

Set of output ports:

$ROP = \{srop, mop\}$ where

srop: output port for emitting events to successor responsibilities.

mop: output port for emitting measures to evaluate quality attributes (performance).

Set of values for *srop*:

- *activated*: this value is emitted when the execution of the responsibility has been started, it is ready for the execution.
- *finished*: the responsibility has been carried out, its execution has been finished.

$Y_{RM,srop} = \{activated, finished\}$

Set of values for *mop*:

$Y_{RM,mop} = R_0^+$ it indicates the time that a responsibility needs to be executed.

Set of output ports and output values pairs:

$Y_{RM} = \{(srop, activated), (srop, finished), (mop, m)\}$ where $m \in Y_{RM,mop}; srop, mop \in ROP$

3.1.4 Internal Transition Function

The internal transition function defines the next state of the responsibility, as result of the elapsed time without an external event has taken place. The "*active*" transitory state indicates that the responsibility can be carried out, because their predecessors have been finished in a suitable form. It communicates its activated state to the corresponding simulation elements; it has been started. Thus, an internal transition is required to send an event using the corresponding port and it changes automatically its state to "*executing*". The other internal transition happens when the execution time has elapsed, returning to the passive state ("*inactive*"), in standby, waiting for other external event.

$$\delta_{RM,\text{int}}(active, \sigma) = (executing, execution\_time)$$

$$\delta_{RM,\text{int}}(executing, \sigma) = (inactive, \infty)$$

3.1.5 External Transition Function

This function produces a state transition when an external event happens. In other words, this change occurs when "*finished*" value is received from fulfilled responsibilities.

$$\delta_{RM,ext}(phase, \sigma, e, x)\begin{cases}(active, 0) & \text{if } phase = inactive \\ (phase, \sigma - e) & \text{if } phase = \text{ active or executing}\end{cases}$$

3.1.6 Output Function

The output function generates output values and then performs an internal transition of states.

$$\lambda_{RM}(active, \sigma) = (srop, actived)$$

$$\lambda_{RM}(executing, \sigma) = (srop, finished) \wedge (mop, m) \text{ where } m \in Y_{RM,mop}$$

3.1.7 Time Advance Function

This function defines the time that the responsibility has been in a state. For the proposed model: $ta(s) = \sigma$

$ta_{RM}(active, \sigma) = 0$ where s is a transitory state.

$ta_{RM}(executing, \sigma) = execution\_time$ (specified parameter for this model).

$ta_{RM}(inactive, \sigma) = \infty$ where s is a passive state.

3.2 DEVS Model for Simple Architectural Component

The simple component (*SC*) is in charge of a set of responsibilities. The relationship between this element and their responsibilities can be represented as a hierarchy of DEVS models. The instances of *RM* are components of the *SC* coupled model.

$$SC = \left(X_{SC}, Y_{SC}, D_{SC}, \{M_{SC,d} \mid d \in D_{SC}\}, EIC_{SC}, EOC_{SC}, IC_{SC}\right). \quad \textbf{(3)}$$

3.2.1 Set of Input Ports and Input Values (X)

SC, coupled model, has a set of input ports, which propagate the input values to its components (elements of the simulation model).

Set of input ports:

$SCIP = \{peip\}$ where *peip* is the input port that receives information from previous architectural elements, and it is connected to the input ports of the components.

Set of input values:

The set of input values represents information about predecessor architectural elements, the execution end of the previous components, which is needed to activate the execution of the corresponding components.

$X_{SC,peip} = \{finished\}$

Set of input ports and input values pairs:

$X_{SC} = \{(peip, finished)\}$ where $peip \in SCIP$

3.2.2 Set of Output Ports and Output Values (Y)

This set represents events that are generated by the components of this model and propagated to other simulation elements.

Set of output ports:

$SCOP = \{seop, mop\}$ where

seop: output port that emits events to the successors elements.

mop: output port that returns a measure value (indicators of quality attributes).

Set of output values:

Output values for the first port (*seop*):

- *activated*: indicates that the execution of the component has been started.
- *finished*: indicates that the execution of the component has been ended.

$Y_{SC,seop} = \{activated, finished\}$

Output values for the second port (*mop*):

$Y_{SC,mop} = R_0^+$ real number that indicates a measure used to calculate a quality index.

Set of output ports and output values pairs:

$Y_{SC} = \{(seop, activated), (seop, finished), (mop, m)\}$ where $m \in Y_{SC,mop}, seop \in SCOP, mop \in SCOP$

3.2.3 Set of Components (D)

This set (D_{sc}) details the references to the components (*RM*) that are part of the *SC*.

3.2.4 Component (Models)

$M_{SC,d} = \text{RM} \quad \forall d \in D_{SC}$ where *d* is a reference name of a component of the *RM* class.

3.2.5 Couplings

The different types of couplings are expressed by the following expression:

External Input Coupling (EIC):

$$EIC_{SC} \subseteq \{((SC, peip_{SC}), (r, prip_r)) \mid r \in D_{SC}, peip_{SC} \in SCIP, prip_r \in RIP_r\}$$

External Output Coupling (EOC):

$$EOC_{SC} \subseteq \{(((r, srop_r), (SC, seop_{SC})), ((r, mop_r), (SC, mop_{SC}))) \mid r \in D_{SC}; seop_{SC}, mop_{SC} \in SCOP, srop_r, mop_r \in ROP_r\}$$

Internal Coupling (IC):

$$IC_{SC} \subseteq \{((r_1, srop_{r_1}), (r_2, prip_{r_2})) \mid r_1, r_2 \in D_{SC}, srop_{r_1} \in ROP_{r_1}, prip_{r_2} \in RIP_{r_2}\} \text{ where } r_1 \Rightarrow r_2$$

$$((r_1, srop_{r_1}), (r_2, prip_{r_2})) \in IC_{SC} \Rightarrow r_1 \neq r_2 \quad \forall r_1, r_2 \in D_{SC}$$

Observation: DEVS model for the connection mechanism is specified in the same form like *SC*, with the corresponding I/O ports and couplings between *RM* instances.

3.3 Implementation using DEVSJAVA

DEVSJAVA [15] is a set of libraries for implementing DEVS models employing JAVA programming language. The main package, *Zdevs,* contains *devs* class, which is the superclass of *atomic* class and *coupled* class, from which is inherited the *digraph* class. The proposed simulation elements were implemented as subclasses of these: *ResponsibilityM (atomic)*, *SimpleComponentM (digraph)* and *ConectionMechanismM* (*digraph*).

4. Example of Application

A case study, a classic case of software architecture, *Model-View-Controller* pattern (or *MVC*) is described. Fig. 2 shows a high level view of the *MVC* pattern using uses case maps (UCM). The definition of the responsibilities on the map depends on the degree of detail that we want to include. It is important to avoid particularities that can confuse because of the abstraction level that we work in the architectures design [16].

In the case study, the controller is responsible for handling events that happen as a result of user requests (responsibility 1 in Fig. 2). Then, it communicates with the model and updates it according to the requested action (responsibility 2 in Fig. 2). The model is the representation of the information used by the system to operate, and it is responsible for changing it (responsibility 3 en Fig. 2), receiving the modification orders from the controller. The task of displaying the user interface is delegated to the view (responsibility 4 in Fig. 2), which takes the data from the model and generates the interface, displaying the changes performed in the model.

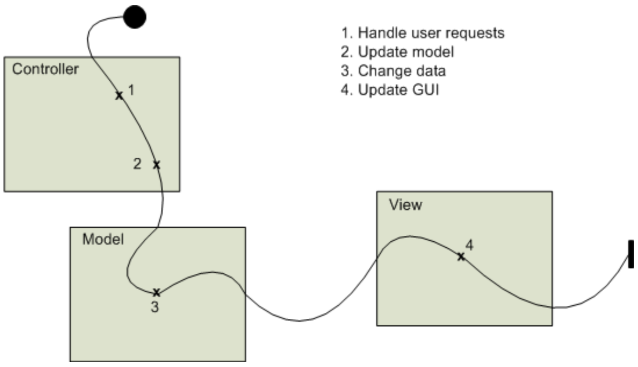


Fig. 2. UCM of *MVC* pattern (adapted from [16])

The three architectural components are traduced into DEVS coupled models (*SC*), and the responsibilities to DEVS atomic models (*RM*), describing how the ports are connected, and how DEVS components are structured for the *Controller* component (*SC1*). Fig. 3 shows how the responsibilities of this component are coupled in the simulation model and the connections of its ports. Each responsibility is an atomic model (*RM*), and it is in charge of sending the corresponding outputs and of calculating values to evaluate architectural aspects (i.e. execution time, considered currently in the model). The coupled model (*SC1*) propagates the information to its components (*rm1*, *rm2*) and to the outside, inputs or outputs respectively.

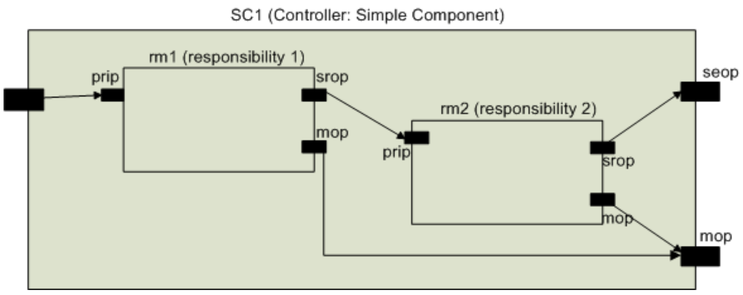


Fig. 3. *Controller* component (*CS*). Ports and internal components (*RM*)

5. Conclusion

In this work, we propose a conceptual model that “captures” information about basic elements of software architecture, the behavior of them, and functional items (responsibilities) added to the structural elements. Additionally, the model represents quantitative values which enable quantitative analysis for validating several scenarios.
To complement the conceptual model, DEVS formalism has been employed to incorporate the advantages of discrete event simulation in the context of the

architectures design. This formalism, unlike other simulation tools, keeps the model uncoupled from simulator, building its elements in modular and hierarchical way. Thus, the transformation of the model concepts (software architecture information) into elements of the simulation model has been described.
As future work, other domain concepts such as composite components and patterns will be study to specify and implement them in the simulation model. Also, quantitative aspects of the architectural models will be study in more detail to incorporate parameters into the simulation elements to evaluate visible attributes at runtime (quality attributes).

References

1. Bass, L.; Clements, P.; Kazman (2003). Software Architecture in Practice. Addison-Wesley.
2. Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R.; Stafford, J. (2002). Documenting Software Architecture- Views and Beyond. Addison-Wesley.
3. Hofmeister, C.; Nord, R.; Soni, D. (2000). Applied Software Architecture. Addison-Wesley.
4. Kruchten. P. (1995). The 4+1 View Model of Architecture. In: IEEE Software, vol. 12, 42-50.
5. Buhr, R. (1999). Making Behaviour a Concrete Architectural Concept. In: 32nd Hawaii International Conference on System Sciences, USA.
6. Buhr, R.; Casselman, R. (1999). Use Case Maps for Object-Oriented Systems. Prentice Hall.
7. Wang, W.; Wu, Y.; Chen, M. (1999). An Architecture-Based Software Reliability Model. In: Pacific Rim International Symposium on Dependable Computing, 143-150.
8. Fukuzawa, K..; Saeki, M. (2002). Evaluating Software Architecture by Coloured Petri Nets. In: 14th International Conference on Software Engineering and Knowledge Engineering, USA.
9. Spitznagel, B.; Garlan, D. (1998). Architecture-Based Performance Analysis. In: 1998 Conference on Software Engineering and Knowledge Engineering.
10. Petriu, D.; Woodside, M. (2002). Software Performance Models from System Scenarios in Use Case Maps. In: 12th International Conference on Computer Performance Evaluation, Modelling Techniques and Tools, UK.
11. Zeigler, B., Praehofer, H., Kim,. T. (2000). Theory of Modeling and Simulation–Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press.
12. ISO/IEC WD3 42010. IEEE P42010/D3. Systems and Software Engineering – Architectural Description (2008).
13. Wirfs-Brock, R.; McKean, A. (2002). Object Design: Roles, Responsibilities and Collaborations. Addison-Wesley.
14. Verbraeck, A.; Valentin, E. (2008). Design Guidelines for Simulation Building Blocks. In: 2008 Winter Simulation Conference (WSC), 923-933.

15. Zeigler, B., Sarjoughian, H. (2005). Introduction to DEVS Modeling an Simulation with JAVA: Developing Component-Based Simulation Models, University of Arizona, USA.
16. Buhr, R.; Casselman, R.; Pearce, T. (1996). Design Patterns with Use Case Maps: A Case Study in Reengineering an Object-Oriented Framework, technical report, SCE 95-17.

Effort Estimation Analysis by Applying Use Case Points

CRISTIAN A. REMÓN[1] AND PABLO THOMAS[2]

[1] Dpto. I+D Maker Electrónica, Mar del Plata, Argentina.
cremon@makerelectronica.com.ar.
[2] III-LIDI, Instituto de Investigación en Informática,
Facultad de Informática, Universidad Nacional de La Plata, Argentina.
pthomas@lidi.info.unlp.edu.ar.

Abstract. *The various effort estimation methodologies in the software development process have arisen due to project failures and successes. The evolution of estimation methods has not been a consequence of their invalidity or erroneous results, but of the evolution of technology itself, of the reduction of error margins and market demands in the search for higher-quality products. This paper shows that the Use Case Points methodology proposed by Frohnhoff and Engels, which clearly proved to be successful during its development process, needs adjustments and/or improvements. In this sense, specific study cases are presented.*

Keywords. *Use Cases, Use Case Points, Effort Estimation.*

1. Introduction

The use of software in society has increased significantly, and therefore, quality software development is required.
Software quality is measured, primarily, through the completion of the requirements defined in the initial stages of the development lifecycle and by the process applied during this lifecycle.
To achieve this goal, the entire software development process must be managed under an effective project plan [1], which ensures that final cost and time are as described in the previously defined requirements.
Software development is an economic activity, and it is therefore subject to monetary restrictions in addition to the inherently technical ones, which are explicitly within the project plan [2].
The activities or tasks that must be carried out to achieve the stipulated product require the investment of effort, which is estimated based on the requirements obtained during the elicitation stage. The elements involved in the estimation of software projects are: size, invested effort, development time, technology used, among others. The invested effort is a key element, since this value, as part of the cost of the project, is used to calculate the profit margin that is obtained with the finished product.
One of the methods used to estimate the software development effort is based on Use Case models [3], which is a widely used technique to describe the

interaction of users with a software system. UML (Unified Modeling Language) designers recommend developers to follow a Use Cases modeled process to be used as input document for the design and the testing stage [4]. Gustav Karner [6] took the Use Cases model to improve the Function Points technique for estimating the effort in software projects.
Effort estimation methodologies lie in two clear objectives: estimating the effort at a preliminary development stage, and relatively reducing error magnitude.
In this article, 12 study cases will be surveyed: 9 graduate dissertations on software projects presented by students of Engineering in Computer Science from the School of Engineering of the FASTA University in the city of Mar del Plata, Argentina, and 3 software industry projects. From these cases, the results obtained are analyzed by applying the enhanced Use Case Points methodology proposed by Frohnhoff and Engels [5], compared with the results obtained in the estimation performed in each of the study cases.
In section 2, the methodology proposed by Gustav Karner for effort estimation is described. In section 3, the adjustments incorporated by Frohnhoff and Engels to Gustav Karner's method are detailed. Then, in section 4, a description is included, together with the values obtained when applying Frohnhoff and Engels' method to the study cases herein proposed. In section 5, an analysis and simulation of Frohnhoff and Engels' method is carried out using the mean values obtained in section 4, with primary focus on the A-factor variable of the method. Finally, in section 6, the conclusions obtained and future works are presented.

2. Gustav Karner's Use Case Points (UCP) Methodology

The use case points methodology is a derivation of the function points methodology proposed by Albrecht. Karner [6] bases this methodology in the utilization of use cases as input data to calculate the effort in the number of man-hours (mh) that are needed for the development of a software project. The effort estimation method uses four main variables:

2.1 Classification of the Actors Involved

The actors involved in the use cases are classified based on their intrinsic characteristic and the way in which they interact with the system. A *simple actor (weight=1)* is that which represents a programming interface or API (e.g., abstraction layer); a *medium actor (weight=2)* is that which interacts through a protocol (e.g., TCP/IP, HTTP, FTP); and a *complex actor (weight=3)* is that which interacts through a graphic interface. Based on this classification, each actor is assigned a value that is known as *weight* (the author of the methodology does not specify the origin of these weights in any of the cases).

2.2 Use Case Classification

Use cases are classified based on the number of transactions they have, including transactions from alternative scenarios and excluding extensions or inclusions of other use cases. A *simple use case (weight=5)* is that with 3 or less transactions; a *medium use case (weight=10)* is that with 4 to 7 transactions; and a *complex use case (weight=15)* is that with more than 7 transactions. Again, each use case is assigned a *weight.*

2.3 Software Project Technical Complexity Factor

Technical factors (T) are defined by the technical influences that may affect the development process of the system to be built. Each technical factor has a complexity degree that ranges between 0 and 5, where 0 means an irrelevant or null value and 5 determines a value with a high degree of influence. Each technical factor has a *weight* value. The total weight of that technical influence factor is obtained by multiplying the assigned complexity value and the weight corresponding to the factor.

2.4 Project Environment Factors

Environment factors (E) indicate the influence of the human group involved in the project on the system to be developed. The same as technical factors, environment factors have an influence degree that ranges between 0 and 5, where 0 means an irrelevant or null value and 5 determines a value with a high degree of influence. Each environment factor has a *weight* value. The total weight of that technical influence factor is obtained by multiplying the assigned influence value and the weight corresponding to the environment factor.

To determine the estimation of the Use Case Points, the following steps must be followed:

1. Classifying actors to determine the UAW (Unadjusted Actor Weight) value
   UAW = Sum of all weights of identified actors
2. Classifying use cases to determine the UUCW (Unadjusted Use Case Weight) value
   UUCW= Sum of all use case weights
3. Determining the UUCP (Unadjusted Use Case Point) value
   $UUCP = UAW + UUCW$
4. Determining the TCF (Technical Complexity Factor) value
   $TCF= 0.6 + (0.01 * \Sigma (T_1...T_{13}))$
5. Determining the EF (Environment Factor) value
   $EF = 1.4 + (-0.03 * \Sigma (E_1...E_8))$
6. Determining the AUCP (Adjusted Use Case Point) value
   $AUCP = UUCP * TFC * EF$

Karner establishes a factor of 20 man-hours for use case point to carry out the estimation stage of a software project [6].

*UCP = AUCP * 20*

The Use Case Point (UCP) value obtained indicates the effort in man-hours that need to be invested to develop the software project.

3. Frohnhoff and Engels Adjustment on the UCP Method

Frohnhoff and Engels performed estimation tests on 15 software projects from various industrial sectors by comparing the effort estimated by applying Karner's UCP method with the real effort of the project, obtaining as a result a standard deviation of 42% [5].

A deviation of such magnitude is not acceptable in the software industry because it tends to overestimate the real effort required.

The authors surveyed the causes for these deviations with the project leaders of the different study cases, and arrived at two main hypotheses:

- Lack of use case standardization
- The software development process model applied:
    - standard processes (Rational Unified Process (RUP), V-model)
    - limited processes
    - processes with strict development policies

By emphasizing the process models hypothesis, Frohnhoff and Engels focused on the adjustments on the influence of requirement definition in the development process and the effort invested in the project. The model proposed for calculating project effort (PE) is the following:

*PE = (PF * M-factor) * (T-factor * A-factor)*

A-factor (Application factor): it is defined by the classification of use cases, which represent project functional requirements. This value is similar to UUCP in the original method.

The complexity level of each use case is standardized in accordance to their main scenarios, transactions and dialogue boxes:

- Simple: <= 3 main scenarios, transactions and dialog boxes (5 points)
- Medium: <= 7 main scenarios, transactions and dialog boxes (10 points)
- Complex: >= 8 main scenarios, transactions and dialog boxes (15 points)

Classification weights are taken based on the original UCP method, n is the number of use cases, resulting in:

A-factor = (Sum of 1 to n of the use cases weight) + UAW (original value of the UCP method)

T-factor: It represents non-functional requirements. It takes the original variables and weights from the UCP method and standardizes complexity influence values. A guide is included to indicate the degree of influence with values 0 (irrelevant), 3 (medium influence), or 5 (high influence). The influence degree is called G, and the weight for the complexity technical factor is W, resulting in:

$T\text{-}factor = 0.58 + \Sigma\ i = 1...13\ (Wi * Gi * 0.01)$

M-factor (Management Factor): The environment influence variables (E) in the original method are reduced from 8 to 6. The E1 factor (knowledge of RUP) and the E3 factor (knowledge of the OO paradigm) are excluded on the grounds that they do not have a significant influence, since there is a high percentage of projects that are developed using UML processes and under OO paradigms. A guide is included to indicate the degree of influence with values 0 (irrelevant), 3 (medium influence), or 5 (high influence).

The authors add a new influence variable, M7, that represents the model of the process to be used, as shown in Table 1.

Table 1. M7 as new variable and its various influence degrees

M-factor	Influence degree	Weight
M7- Process Model	0: limited processes 3: standard processes (RUP, V-model) 5: processes with strict development policies	-4.5

Then, the new M-factor classification according to Frohnhoff and Engels is established as indicated in Table 2.

Table 2. M-factor classification

M-factor	Description	Weight
M1	Experience in the application	-0.5
M2	Project leader skills	0.0
M3	Motivation	2.5
M4	Stable Requirements	2.0
M5	Part-Time developers group	-1.0
M6	Difficulty in programming language	0.0
M7	Process model	-4.5

Finally, considering G as the influence degree and M as the weight of the value of M-factor, the result is:

$M\text{-}Factor = 0.8875 - \Sigma i = 1...7\ (Mi * Gi * 0.025)$

The standardization of M-factor weights caused a scalability adjustment in the productivity factor (PF) from 28.7 to 35 mh/UCP.

Frohnhoff and Engels applied the new, adjusted method on the 15 projects mentioned at the beginning of this section, and obtained as a result a deviation of 20% between the estimated effort and the real effort of the project; in contrast with the 42% deviation obtained using Karner's method [5].

4. Application in Study Cases

The UCP method adjusted by Frohnhoff and Engels was applied in 12 study cases – 9 graduate dissertations on software projects (limited processes according to the M-factor classification) from the School of Engineering of the FASTA University in the city of Mar del Plata, Argentina, and 3 software industry processes from the company Maker Electrónica, Argentina (www.makerelectronica.com.ar).
In these study cases, the same development policies were applied (standard processes based on the M-factor classification). The development effort required was estimated using the methodology proposed by Frohnhoff and Engels, as well as the experience in the field of the project manager and the group of developers, and interesting conclusions were drawn after the end of the projects.

4.1 Project Description

Below, the projects used are briefly described:

- CE1- Current account management in limo cooperative: Computer system to manage current accounts and information of the owners of cars under contract with the cooperative.
- CE2- Technical Management and Dashboard of AVL equipment: Design and implementation of a software system that allows managing all relevant information and lifecycle of AVL (Automatic Vehicles Localization) satellite positioning equipment produced by the company Maker Electrónica. Information centralization and unification of the current client Business Management system.
- CE3- AVL-Desktop: Design and implementation of a desktop system that displays the position of the company vehicle fleet on a digital map in real time. The satellite monitoring of mobiles is performed through a Web application; the AVL-Desktop project is an extension of the Web AVL application.
- CE4-GAMA System [9]: Computer system that helps manage ambulatory medical care of patients at the National Epidemiology Institute (INE) of Mar del Plata, Argentina, applicable to other health institutions.
- CE5- SCRUM [10]: Computer solution for the Rugby Union of Mar del Plata, Argentina, for the comprehensive management of their activities, through the development of a Web application and a desktop application.
- CE6- DO-RE-MI [11]: Management system for a music conservatory.

- CE7- SLT [12]: The main purpose of the project is providing end users with a computer solution that manages in a comprehensive and efficient manner the logistics of a transportation company.
- CE8- SIARER [13]: Computer system for the analysis of energy resources.
- CE9- MARATHON [14]: Sports management and training planning system.
- CE10- ZONDA [15]: Computer system for the analysis of renewable energy resources.
- CE11-C.T.E.N.P [16]: Computer system for managing the activities of the National Fishing School of the city of Mar del Plata, Argentina.
- CE12-GySP [17]: Computer system for managing and monitoring software projects.

4.2 Results obtained

The values obtained with Frohnhoff and Engels' method are presented in Table 3. Effort estimation values for the study cases are expressed in man-hours.

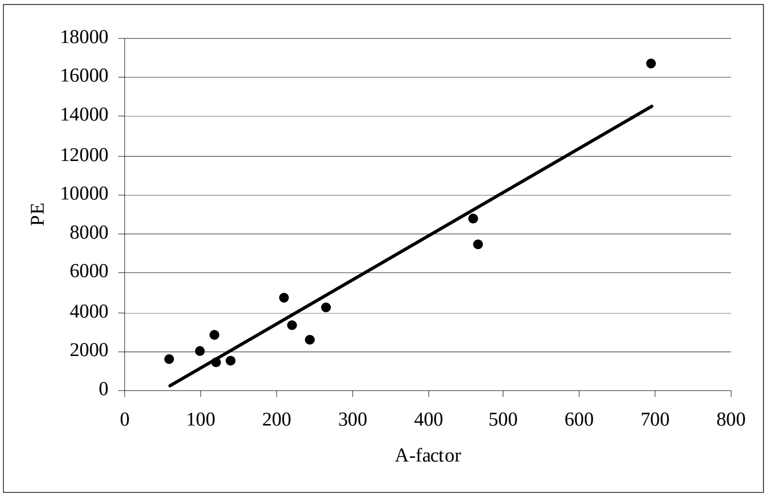
Table 3. Values obtained with Frohnhoff and Engels' method

Study Case	Project Type	A: Real Effort (MH)	A-factor	T-factor	M-factor	B: PE(MH)	Gap (B-A) /A*100
CE01	Management	750	211	0.915	0.700	4730.09	530.68%
CE02	Industry	173	120	0.800	0.825	2772.00	1502.31%
CE03	Industry	305	59	0.910	0.850	1597.28	423.70%
CE04	Management	5085	461	0.940	0.575	8720.97	71.50%
CE05	Management	2025	266	0.925	0.488	4198.23	107.32%
CE06	Management	1236	696	0.980	0.700	16710.96	1252.02%
CE07	Management	3762	468	1.070	0.425	7448.81	98.00%
CE08	Simulation	4577	101	1.005	0.550	1953.97	-57.31%
CE09	Management	1810	246	0.905	0.325	2532.42	39.91%
CE10	Simulation	3282	140	0.940	0.325	1496.95	-54.39%
CE11	Management	1244	221	0.940	0.450	3271.91	163.015
CE12	Management	1641	121	1.015	0.325	1397.02	-14.87%
Mean		*2157.50*	*259.17*	*0.95*	*0.54*	*4671.98*	*116.55%*

5. Analysis of A-factor

Figure 1 analyzes the relation between Project Effort (PE) and the value of A-factor, and it shows that PE value tends to increase as the value of A-factor increases in minimum ranges.

Fig. 1. Linear trend of Project Effort in relation to the value of A-factor



To exemplify this trend, the mean values of the study cases used are considered.
*PE=(35mh * M-factor)*(T-factor * A-factor) = (35mh * 0.54) * (0.95 * 259)= 4649mh*

The mean values from Table 3 are used below to demonstrate, through a specific example, the sensitivity of the A-factor variable in relation to project context; the result obtained is an overestimation of the effort required.
If a new use case of medium complexity is added, it receives the corresponding weight of 10, and the equation for calculating the effort would then be:
*PE= (35mh *M-factor)*(T-factor * A-factor)=(35mh * 0.325)* (0.905 * 269) =4819mh*

The introduction of a new functionality to the project increased the estimated effort in 3.6%, with a difference of 170mh. This means that the new medium-complexity functionality requires the investment of 170mh.
The sensitivity of the A-factor value can be observed through the inclusion of a medium-complexity requirement, which resulted in an overestimation that is beyond the acceptable dispersion margins, optimistic or pessimistic, in relation to the estimation of the effort required for the project.

The same as Frohnhoff and Engels analyzed Gustav Karner's method and proposed an improvement by focusing in the technological and environmental variables, the analyses presented in this article suggest an improvement on that method by focusing on the A-factor variable, which is sensitive to the context of the functional requirements in a software project.
The improvement proposed is to achieve a normalization of use cases that derives their qualitative and quantitative reclassification and thus strengthen the influence of the A-factor variable.

6. Conclusions and Future Work

The discussion in this article indicates that the application of the Use Case Points proposed by Frohnhoff and Engels on twelve real study cases generates excessive deviation in the estimation of the effort required for a software project due to the sensitivity of the A-factor variable. This sensitivity was demonstrated through a specific example in section 5.
The quantitative and qualitative classification given by Frohnhoff and Engels to use cases is not a convincing parameter when estimating the effort of a preliminary stage of the project, since it is the cause for the weakness of the A-factor variable in this method. As a consequence of this characteristic and the results shown in this paper, an adjustment of this variable is required.
The same as Frohnhoff and Engels focused their improvement on technological and environmental aspects, this article suggests applying an improvement to the environment of the functional requirements of the project, taking as starting point the normalization of use cases, that is the input information of this variable, to then tackle the reclassification of use case complexity.
Use cases are a useful and important tool in software project development. They represent the starting point for the modeling of an object-oriented system, the baseline points for project planning, and the input documentation in requirement validation and testing stages.
Finally, the following future lines of work are planned:

- Normalization of the use cases used in order reclassify them based on their complexity.
- Analysis of the values and influence degree of the A-factor variable for each project.
- Development of new field tests applying the adjustments.

References

1. Kusumoto, S., Matsukawa, F., Inoue, K., Hanabusa, S., Maegawa, Y. (2004). Estimating Effort by Use Case Points: Method, Tool and Case Study. In: Proceedings of the 10th International Symposium on Software Metrics, 292-299.

2. Sommerville, I. (2002). Ingeniería de Software, 6ta. Edición. Addison, Wesley.
3. Anda, B., Dreiem, H., Sjøberg, D.I.K., Jørgensen, M. (2001). "Estimating Software Development Effort Based on Use Case – Experience from Industry. En: M. Gogolla, C., Kobryn, C. (Eds.) UML 2001. LNCS, vol. 2185, 487-502. Springer-Verlag.
4. Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G. (1992). Object Oriented Software Engineering: A Use Case Driven Approach. (ACM Press) Addison-Wesley.
5. Frohnhoff, S., Engels, G. (2008). Revised Use Case Point Method - Effort Estimation in Development Projects for Business Applications. In: Proceedings of the 11th International Conference on Quality Engineering in Software Technology, Potsdam, Dpunkt-Verlag.
6. Karner, G. (1993). Metrics for Objectory, Degree thesis, University of Linkoping, Sweden.
7. Coleman, D. (1998). A Use Case Template: draft for discussion. Fusion Newsletter.
8. Cockburn, A. (2001). Writing Effective Use Case. Boston, Addison-Wesley.
9. Cucchi Colleoni, A., Di Crocce, V., Sansevero, R. (2007). Thesis Project: GAMA. Facultad de Ingeniería, Universidad FASTA, Mar del Plata.
10. Ghigliani, J., Fernández, G. (2005). Thesis Project: SCRUM. Facultad de Ingeniería, Universidad FASTA, Mar del Plata.
11. Cosia, V., Villen, C. (2009). Thesis Project: DO.RE.MI. Facultad de Ingeniería, Universidad FASTA, Mar del Plata.
12. Ferrari, M., Ortiz, S., Seoane, L., Ullo, M. (2007). Thesis Project: SLT. Facultad de Ingeniería, Universidad FASTA, Mar del Plata.
13. Abadie, E., Bressán, J., Guzmán, E. (2010). Thesis Project: SIARER. Facultad de Ingeniería, Universidad FASTA, Mar del Plata.
14. Gáspari, F., Remón, C. (2006). Thesis Project: MARATHON. Facultad de Ingeniería, Universidad FASTA, Mar del Plata.
15. Albornoz, D., Posse, J., Speratti, N. (2010). Thesis Project: ZONDA. Facultad de Ingeniería, Universidad FASTA, Mar del Plata.
16. Aroca, A., D'Angelo, J. (2007). Thesis Project: CTENP. Facultad de Ingeniería, Universidad FASTA, Mar del Plata.
17. Pavón, J., Rueda, J. (2006). Thesis Project: GySP. Facultad de Ingeniería, Universidad FASTA, Mar del Plata.

The Importance of Using Empirical Evidence in Software Engineering

ENRIQUE FERNÁNDEZ[1], OSCAR DIESTE[2], PATRICIA PESADO[3] AND RAMÓN GARCÍA-MARTÍNEZ[4]

[1] PhD Program on Computer Sc. School of Computer Sc. Universidad Nacional de La Plata.
[2] Empirical Software Eng. Group. School of Computer Sc. Universidad Politécnica de Madrid.
[3] Instituto de Investigaciones en Informática LIDI. Facultad de Informática. UNLP – CIC.
[4] Information Systems Research Group. Productive & Technologic Development Dept. Universidad Nacional de Lanús.
enriquefernandez@educ.ar, odieste@fi.upm.es, ppesado@lidi.info.unlp.edu.ar, rgarcia@unla.edu.ar.

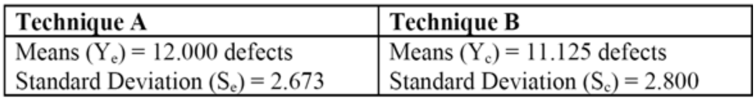
Abstract. *Experiments that are run with few experimental subjects are often considered not to be very reliable and deemed, as a result, to be useless with a view to generating new knowledge. This belief is not, however, entirely correct. Today we have tools, such as meta-analysis, that we can use to aggregate small-scale experiments and output results that are equivalent to experiments run on large samples that are therefore reliable. The application of meta-analysis can overcome some of the obstacles that we come up against when running software engineering experiments (such as, for example, the practitioner availability problem).*

Keywords. *Meta-analysis, statistical power, reliability, replications, sample size.*

1. Introduction

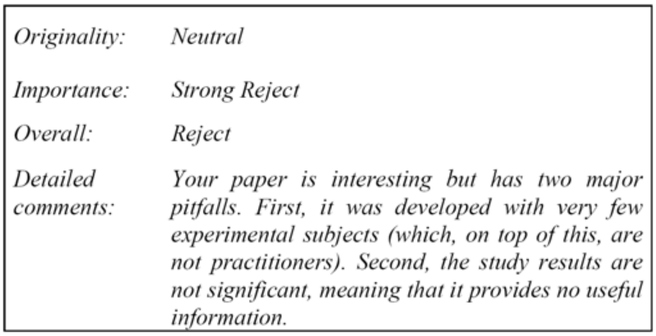
Suppose that a hypothetical Dr. Smith is a university researcher working on testing techniques [1]. Recently, Dr. Smith has read about new inspection technique "A" that looks as if it might outperform other techniques, like, for example, technique "B". And so, he decides to run an empirical study to test this hypothesis. To do his, he puts out a call for final-year BSc in Software Engineering students to participate in the study. As a result of the all, he manages to recruit 16 students, and 8 are trained in the new technique and the other 8 in the pre-existing technique. During the experiment, each group applies the respective technique to the same program. He measures the number of effects detected as the response variable. Table 1 shows the results (aggregated by group).

Table 1. Results of the experimental study by Dr. Smith



Technique A	**Technique B**
Means (Y_e) = 12.000 defects Standard Deviation (S_e) = 2.673	Means (Y_c) = 11.125 defects Standard Deviation (S_c) = 2.800

Based on these values, Dr. Smith runs a hypothesis test (a t-test assuming variances to be equal) with $\alpha = 0.05$. This test returns a p-value of 0.53. Therefore, technique A cannot be said to perform better than B.
Although the results are not promising, Dr. Smith decides to go ahead with their publication in the hope that the experiment will be replicated and the aggregation of data will better explain the comparison between A and B. Dr. Smith submits the paper and, at the end of the review process, receives the assessment shown in Figure 1.



Originality:	*Neutral*
Importance:	*Strong Reject*
Overall:	*Reject*
Detailed comments:	*Your paper is interesting but has two major pitfalls. First, it was developed with very few experimental subjects (which, on top of this, are not practitioners). Second, the study results are not significant, meaning that it provides no useful information.*

Fig. 1. Results of the paper review process

The above example, albeit fictitious, is representative of many real pieces of empirical software engineering (ESE) research. On the one hand, many researchers interpret hypothesis testing too restrictively (and wrong in many cases, as will be seen later), focusing on whether or not the results are significant (level $\alpha = 0.05$). On the other hand, there is a tendency not to take experimental studies that were built with students as evidence, as this research is not considered to be extrapolable to real-world environments. However, there is a shortage of subjects (be they practitioners or students) that are willing to participate in experimental studies [2][3][4]. Additionally, the more subjects an experiment has, the more costly it will be in terms of workload, infrastructure, among others, and this can discourage researchers. On the other hand, the cost of experiments run with fewer subjects is likely to be more affordable. These factors clearly limit SE researchers' prospects of being able to generate new empirically validated knowledge.
Fortunately, there are some alternatives for exploiting the results of small-scale studies. In this paper we will focus on one: meta-analysis. Essentially, meta-analysis is a statistical technique for aggregating more than one study, thereby increasing the number of experimental subjects involved in the hypothesis testing and outputting more reliable results. In our research we have analyzed whether meta-analysis could be applied in ESE to combine the

results of several small-scale experiments, with the aim of increasing the power of experiments with small samples.

We will proceed as follows. Section 2 will describe how sample size affects hypothesis testing. In Section 3 we will outline how to use meta-analysis to combine the results of more than one small study and thus increase their power. Section 4 presents a set of problems related to meta-analysys. Finally, Section 5 will discuss whether meta-analysis is reliable when applied to ESE.

2. State of the Art

Any statistical test is subject to two types of errors: α, or type I error, and β, or type II error [5]. These errors occur due to the uncertainty associated with estimating population parameters (means and standard deviation) from a sample of the population. Remember that an experiment observes what happens in a sample (the subjects that tested the techniques) to estimate what happens in a population (the reality of the tested techniques). As Table 2 shows, α is the error associated with the alternative hypothesis (H_1: there is a difference between tested techniques) being accepted when the null hypothesis (H_0: there is no difference between the tested techniques) holds for the population, and β is the likelihood associated with the opposite event.

Table 2. Decision of the statistical test

	H_0	H_1
H_0	Correct decision (1-α)	β (Tipe II error)
H_1	α (Tipe I error)	Correct decision (1-β)

It is more dangerous for an experiment to lead to the belief that there actually is a difference between two tested techniques when there really is none (error α) than to believe that there is no difference (because none is observed in the experimental sample) when there really is (error β). Therefore, the value of α is set at extremely low values, such as 0.1, 0.05 or even 0.01 (10%, 5% and 1%, respectively).

Unfortunately, α and β are not independent: according to statistical theory, a hypothesis test is characterized by five factors [6]: α, β, the mean difference *d*, the level of variation of the response variable *s* (measured as the variance or standard deviation) and the number of experimental subjects, or, to be more precise, sample size, *n*. Equation 1 shows the relationship between these factors, where z represents the typified normal distribution:

$$z_{1-\beta} = \sqrt{\frac{n}{2}}\frac{d}{S} - z_{1-\alpha} \qquad (1)$$

These five factors form a closed system. This means that an increase or decrease in any one of the factors leads to increases or decreases in the others. In practice, the factor that really is affected is *n*, as type I (α) and type II (β) errors are set beforehand, and both *d* and *s* are circumscribed by the experimental context and cannot therefore be manipulated at liberty by the researcher. This is perhaps the most important, albeit not the only, reason why experiments are required to have a large number of experimental subjects. When the number of experimental subjects is small and α is set at 0.05, β returns very high values.
Let us go back to the example of Dr. Smith's experiment. Applying Equation 1 we get β = 0.83, that is, the test will detect significant differences 17% of the time, whereas it will fail to do so in 83% of the cases, even though they possibly do exist in the population/reality. The influence of the number of subjects on type II error is even clearer if we look at how β decreases as more experimental subjects join, all other factors being equal. It is usual practice to use the term *reliability* instead of α to refer to type I error and *statistical power* instead of β to refer to type II error. Reliability is calculated as 1 - α and power as 1 - β. For an experiment to be considered reliable, it is usual to set type I error at α = 0.05 (that is, a reliability of 0.95 or 95%) and type II error at β = 0.2 (that is, a power of 0.8 or 80%). As Figure 2 shows, Dr. Smith would have needed a total of 120 subjects (60 in each group) for her experiment to be considered reliable. Fortunately, there are several strategies designed to overcome the problems of low power caused by the use of experiments that have few experimental subjects. In this paper, we will look at meta-analysis.

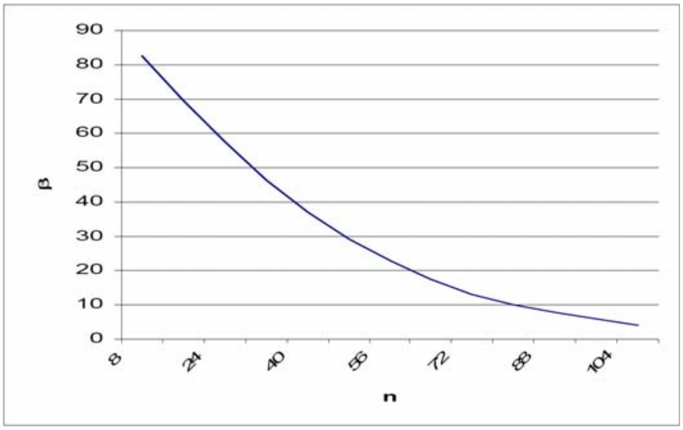


Fig. 2. Decrease in type II error against the increase in n

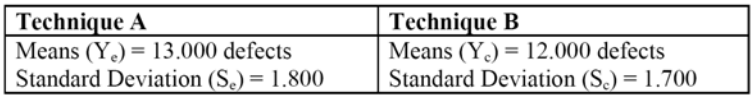
3. Taking Advantage of Experiments with Few Subjects

Meta-analysis is a statistical technique for combining the results of more than one experiment developed previously to achieve a greater statistical power than any of the individual experiments on their own. Although usually associated with medicine, the term meta-analysis as it is now known was developed in psychology.

In many cases of psychology, the treatments studied have very small effects on experimental subjects, meaning, as illustrated in Figure 2 [7], that experiments need a very large sample size (usual guidelines are around 150 [8]). In many cases, however, not that many subjects are available for experiments and studies reporting insignificant effects predominate over others that do detect significant effects, as studies of low statistical power accumulate. This was the way things were in psychotherapy, the specialized field with which Dr. G.V. Glass [9], creator of meta-analysis as we know it today, was concerned. Using an argument very similar to the one brandished in ESE today (small studies are useless), psychotherapy was judged to be ineffective. Dr. Glass, who did not agree with this interpretation, took a different road to demonstrate his belief: instead of excluding studies (on the grounds of their size or statistical significance), he tried to consider as many studies as possible upon which to base his findings. Looking back, the hardest thing was to find a way of aligning the wide range of metrics used in the different replications to measure the response variables. The solution was to come up with what is today the well-known concept of effect size, briefly mentioned in Section 2. Effect size is a non-scalar measure calculated as the difference between the treatment means divided by the pooled standard deviation.

After calculating the effect size of each experiment, all Glass had to do was average the results of the individual experiments to arrive at a *global effect* using a procedure dating back to the mid-19th century[10]. This value represents the effect that, theoretically, a single experiment having a greater sample size and, consequently, a smaller type II error than any of the original experiments would have achieved. This way he demonstrated that psychotherapy was indeed effective. The parallelisms with ESE, in respect of the potential contribution of small studies, are evident. For example, suppose that Dr. Smith published her paper on her laboratory web site. Later Dr. Thomas visited the web site, found the experiment interesting and decided to replicate it. In this case, Dr. Thomas managed to recruit no more than eight advanced MSc in Software Engineering students, four of which he assigned to each of the experimental groups. Results are shown in Table 3.

Table 3. Results of Dr. Thomas' experimental study



Technique A	Technique B
Means (Y_e) = 13.000 defects Standard Deviation (S_e) = 1.800	Means (Y_c) = 12.000 defects Standard Deviation (S_c) = 1.700

Dr. Thomas ran a t-test on these results (assuming variances to be equal at $\alpha = 0.05$) and also found insignificant differences (pvalue = 0.57). What would happen if these two studies were combined using meta-analysis to achieve a new result? Would the differences be significant? Would the test be more powerful? In response to these questions, the sample size is still not big

enough to return significant results (there are only 12 subjects per technique). Figure 3 (showing the statistical power of the meta-analysis for a population with an effect size 2 of 0.5 and α=0.05) indicates that about 70 experimental subjects would be necessary for a meta-analysis to achieve what is usually considered as a discriminative statistical power (1-β = 0.8).
The statistical power, however, has improved in part. Whereas Dr. Smith's and Dr. Thomas' tests had a power of 0.17 and 0.11, respectively, the meta-analysis achieved a power of 0.13. Note that if it had been possible to use 8 + 4 = 12 subjects per group in a single experiment, it would have been possible to achieve a power of 0.22. Using meta-analysis it is possible to gradually increase the statistical power as more experiments are added. This way experiments with a small sample size can supplement each other. The more experiments (no matter how small the number of subjects per experiment is) that are aggregated using meta-analysis, the more powerful the results and, consequently, the greater the possibility of detecting false-negatives will be.

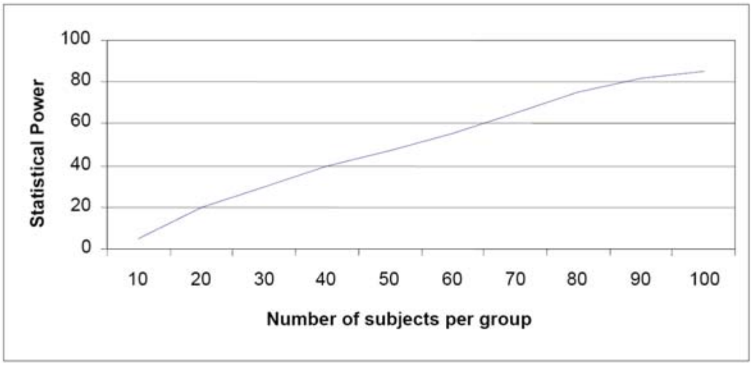


Fig. 3. Increase of the statistical power in a meta-analysis

Suppose that there are three more replications of Dr. Smith's research, whose results are shown in Table 4 (note that they all return insignificant results).

Table 4. Results of Identified Studies (Replications)

Study	Ne	Me	Se	Nc	Mc	Sc	p-value	Power
3	9.00	11.00	1.80	9.00	10.10	1.70	0.30	17.58
4	10.00	10.00	1.40	10.00	9.10	1.50	0.20	20.34
5	12.00	9.00	1.60	12.00	8.10	1.80	0.20	24.63

Figure 4 charts how the power of the meta-analysis increases as more of these studies are added. In this example, even though the test fails to achieve the desired power level of 80%, it does, in any case, manage to output significant differences at a power of almost 57% (which is much greater than the best experiment separately, estimated at 24%).

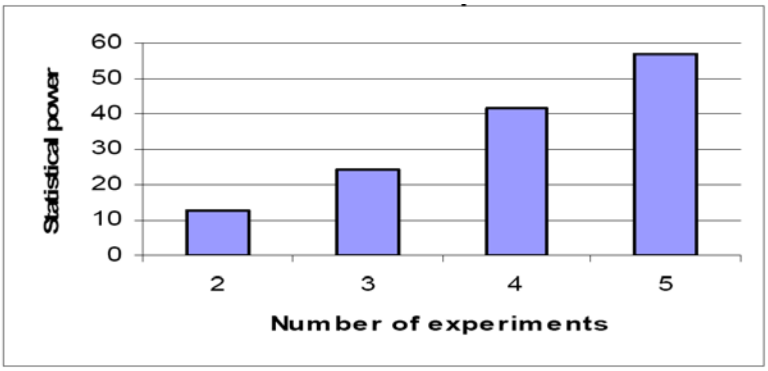


Fig. 4. Increase in the statistical power by accumulating small-scale replications

This is noteworthy, as the example was designed based on the fact that the inspection technique efficiency actually IS different.
So far we have given an example of how meta-analysis can be used to take advantage of studies with small sample sizes that, separately, return results that are insignificant but, together, could provide valuable evidence.

4. Theoretical Model Limitations

Although the functions of estimating the power of a meta-analysis to estimate accurately the statistical power of a meta-analysis when the set of experiments that are part of the aggregation process are homogeneous (the differences between the results of different experiments are minimal), in our view, this function has shortcomings to be applied in the current context of the SE. That problem is that such studies are small, variations in the results, in general, are great influences of experimental error. When this happens the power of meta-analysis tends to decline as indicated obliquely by Hedges and Olkin [11]. We have analyzed this aspect using a Monte Carlo test, which simulate the results of studies within the same population, but without forcing homogeneity among the groups to add, varying the number of subjects for experiments between 4 and 20 and combining 2 to 10 experiments for meta-analysis.
This study allowed us to determine when it has no homogeneity, that for low effect sizes (0.2) is almost impossible to get the test showed significant differences working with 200 subjects per group (simulated maximum sample size) due to the low power statistical effects that for effect sizes medium and high (0.5 and 0.8) heterogeneity is not determinant, as it has been able to achieve good statistical power with lots of subjects no too high (about 80 to 30 subjects per group, respectively). Figure 5 shows a comparison of the estimated power for each of the values of typical effect.

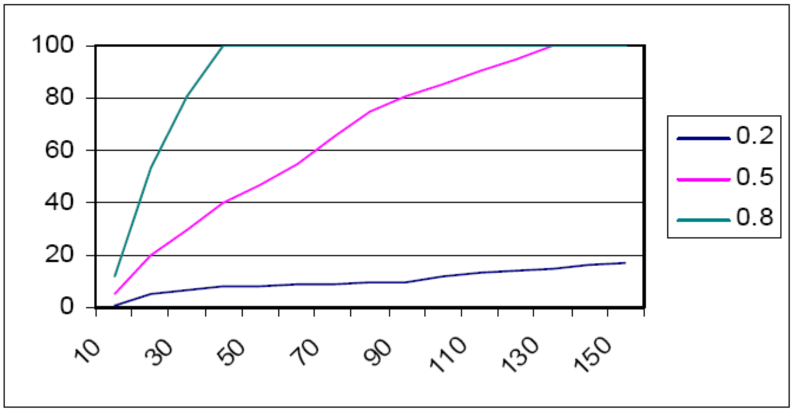


Fig. 5. Simulated power for a meta-analysis

5. Conclusions

In this paper we have shown that there are options open to researchers to generate pieces of empirical SE knowledge more efficiently than they do today. We have shown that meta-analysis is able to increase the power of experiments, enabling a set of small studies that individually do not return statistically significant differences to do so, if taken together. This way we can solve some of the problems related to the accumulation of a sizeable number of experimental subjects by a single researcher, as we can put together a large-scale experiment by meta-analyzing replications of small studies.
In summary, we can say that:

[1] It is worthwhile running experiments even if they do not have many experimental subjects, as they can be combined to form a larger scale study;

[2] It is worthwhile publishing studies even if they do not return significant results, as this can be very often due to the low power of the statistical method.

[3] If this strategy were applied to really important technologies in SE (i.e.: UML or partition of equivalence), the combined effort of the investigators would allow to decide whether these technologies are really useful or not, providing the necessary foundation to become software development in an engineering process.

Acknowledgments

This research has been partially funded by grant UNLa-SCyT-33081 of the National University of Lanus (Argentine) and by grants TIN2008-00555 and HD2008-00046 of the Spanish Ministry of Science and Innovation (Spain).

References

1. Basili, V. R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sörumgård, S., Zelkowitz, M.; 1996; *The empirical investigation of perspective-based reading*, International Journal on Empirical Software Engineering, Vol. 1, No. 2, 133-164.
2. N. Juristo, A. Moreno (2001). Basics of Software Engineering Experimentation, Kluwer Academic Publishers.
3. Tonella P., Torchiano M., Du Bois B., Systä T. (2007). *Empirical studies in reverse engineering: state of the art and future trends*; Empir Software Eng 12:551-571.
4. Dyba, T., Aricholm, E.; Sjoberg, D.; Hannay J.; Shull, F. (2007). Are two heads better than one? On the effectiveness of pair programming. IEEE Software, 12-15.
5. Everitt, B. (2003). *The Cambridge Dictionary of Statistics*, CUP.
6. Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences.* (2nd ed.).
7. Gurevitch, J. and Hedges, L. (20019. *Meta-analysis: Combining results of independent experiments.* Design and Analysis of Ecological Experiments (eds S.M. Scheiner and J. Gurevitch), 347-369. Oxford University Press, Oxford.
8. Fisher RA (1925). *Statistical Methods for Research Workers* (first ed.). Edinburgh: Oliver & Boyd.
9. Glass, G. (1976). Primary, secondary, and meta-analysis of research. Educational Researcher 5: 3-8.
10. Cochrane (2008). Curso Avanzado de Revisiones Sistemáticas; www.cochrane.es/?q=es/ node/198.
11. Hedges, L.; Olkin, I. (1985). Statistical methods for meta-analysis. Academic Press.

VII

Database and Data Mining Workshop

A UML Profile for Fuzzy Multidimensional Data Models

IVÁN RODRÍGUEZ[1] AND JOSÉ LUIS MARTÍ[2]

[1] Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso, Av. Brasil 2147 – Valparaíso, Chile. ivanrodriguezcontreras@gmail.com.
[2] Departamento de Informática, Universidad Técnica Federico Santa María, Av. Vicuña Mackenna 3939 – San Joaquín Vitacura, Chile. jmarti@inf.utfsm.cl.

Abstract. *Over the last several years, multidimensional data modeling has had several proposals for its formalisation; on the other hand, the incorporation of fuzzy logic in databases has increased the need to represent uncertainty. However, to our knowledge, so far projects in both areas have not been developed.*
This paper suggests joining those two needs to create a solution; proposing a UML profile oriented to design multidimensional data models with the presence of fuzzy elements.

Keywords: *Multidimensional Database, Fuzzy Logic, UML Profile.*

1. Introduction

A multidimensional database (MDB) manipulates its data as a (hyper)cube, which is composed of dimensions and facts. A dimension represents a variable of interest to the analysis, while the facts represent the subject (interesting patron, event) that must be analyzed to understand its behavior. For example, for a sale (the fact), it will be important to analyze place, client, time, and products (the dimensions) involved in it, and also to try to understand its relationships. The facts are composed of measures that describe the properties the user wants to optimize. Measures are normally shown quantitatively (additive). Generally, a measure has two parts: a numerical property of the fact (i.e.: price of sale, return) and a formula associated to a simple aggregation function (i.e.: sum, count). For a formal definition, see [1].
For the generation of any data models, there are several graphical notations. UML has become the standard language for this type of task [2]. The fact that UML is a general-purpose language provides great flexibility and expressivity when modeling systems. Nevertheless, there are numerous situations in which it is better to count on more specific language to model and to represent the concepts of certain particular domains. This happens, for example, when the syntax or the UML semantics do not allow expression of the specific domain concepts, or when the author wants to restrict or specialize the UML constructors themselves, which are rather generic.

OMG defines two possibilities for achieving a specific domain language: develop a new language (alternative to the UML); or extend the UML itself specializing some of its concepts and limiting others, but respecting the original semantics of the basic elements (class, attributes, relationships, operations, transitions and so on). This second form is called UML profile; a profile is defined through three mechanisms: *stereotypes*, *constraints*, and *tagged values*. Several UML profiles have been proposed (i.e., real-time systems [3], business modeling [4]); to our knowledge, the only profile that exists to construct conceptual multidimensional data models [5] is based on (crisp) traditional data types.
In all the known cases, the type of data considered is precise. However, today Fuzzy Logic has become an important technique for including imprecise and uncertain concepts, in a way similar to how the human brain works: evaluating many options and then weighing them to make a final decision. Boolean Logic assigns a value *true* (1) when something matches a condition (1), or *false* (0) in any other case. On the other hand, Fuzzy Logic permits relative grades of matching the condition in the inclusive interval [0,1]. Thus, fuzzy sets are a convenient way to represent concepts without precise limits (e.g., high/low temperature).

The general objective of this work is to provide a UML profile that, including fuzzy concepts, can be utilized to generate multidimensional data models.

2. State of the Art

This paper uses two fundamental concepts: the UML profiles to multidimensional data models and fuzzy data models used to design databases. What follows is a short description of both of these.

2.1 UML Profile for a Multidimensional Data Conceptual Model

To our knowledge, only one UML profile has been proposed so far for multidimensional conceptual modeling [5] (for simplicity, this profile is called MDC). In general terms, this profile is defined with the following schema:

- Description: short explanation in natural language.
- Extensions (profiles) or prerequisites: establishes if the proposal needs other extensions for its complete definition.
- Stereotypes: the concrete definition of the elements of the profile.
- Rules: the semantics of the classes is given by a set of invariants defined through OCL.
- Comments: any additional comment, note or example serving to best explain the ideas, usually written in natural language.

In the MDC, the information is structured in facts and dimensions. It is allowed the use of “many-to-many” relationships, but specific attributes must be set to allow this association. Attributes created in this way are called *DegeneratedFacts*. There are some cases in which a dimension is not considered explicitly, given that it is assumed that most of its properties are represented by means of other elements (made or dimensions); nevertheless, some attributes are still needed to identify instances of facts solely. These dimensions are called *degenerated dimensions*.
In this profile, the stereotypes and values tagged are specified by means of a structured scheme; part of the definition of the *Fact* element is given in Table 1 as an example. It is important to mention that the correct use of this profile is assured by the definition of 51 restrictions specified in natural language and expressions OCL [5].

Table 1. Example of definition of the *Fact* element in the MDC Profile (obtained from [5])

Name	**Fact**
Base Class	Class
Description	Classes of this stereotype represent facts in a multidimensional model
Icon	▦
Constraints	All attributes of a fact must be DegenerateDimension or factAttribute: self.feature → select(fe | fe.ocllsKindOf(Attribute)) → forAll(f | f.ocllsTypeOf(DegenerateDimension) or f.ocllsTypeOf(FactAttribute)) …
Labels	None

The authors also present a metamodel for their proposal, which is composed of three levels. Level 1 corresponds to the definition of the model; a package represents a star schema of a multidimensional conceptual model. A dependency between two packages of this level indicates that the star schemas share at least one dimension. Level 2 is the definition of the star schemas; a package represents a fact or a dimension of the schema; a dependency between two packages of a dimension indicates that the packages share at least one level in a hierarchy of dimensions (see the graphical representation in figure 1a). Finally, level 3 corresponds to the definition of facts and dimensions; a package is divided into a set of classes that represent the levels of hierarchy in a dimension package, or the complete star schemas in the case of a fact package (see figure 1b).

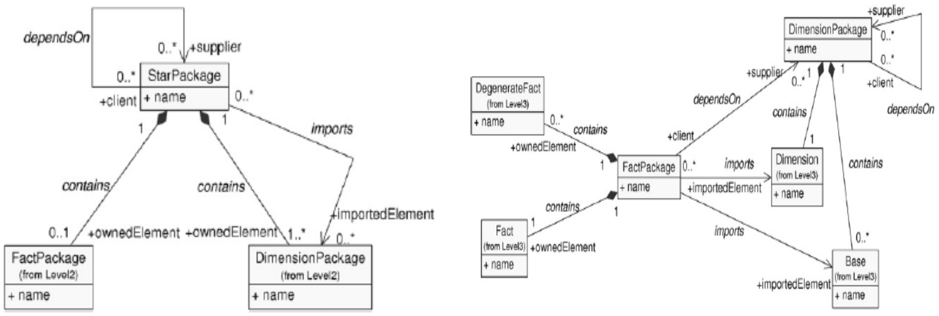


Fig. 1: Level 1 (a) and level 2 (b) of the MDC metamodel (obtained from [5])

2.2 Fuzziness in the Data Modeling

An extension of the Entity-Relationship Model to incorporate fuzziness was proposed in [6], where fuzzy elements as entities, attributes and associations are represented graphically. This extension defines three levels of fuzziness:

1. Entities, attributes and associations could be fuzzy; in other words, could have membership grades to the E-R model.
2. Fuzzy occurrences of entities and associations.
3. Fuzzy values of the attributes, expressed through a distribution of possibility.

In the case of UML, the fuzzy data modeling corresponds to an extension of the class diagram [6,7]. A class could be extended considering:

1. Some objects are fuzzy and share the same properties; therefore, the class defining them is fuzzy. Then, the objects belong to that class with a grade of membership in the interval [0,1].
2. If the domain of an attribute could be fuzzy; then the class including it is fuzzy, too.
3. The subclasses generated from a fuzzy class by specialization, and the superclass produced by one or more fuzzy subclasses through generalization are fuzzy, too.

Figure 2 shows a fuzzy class, which is distinct of the classic case for its discontinuous line. Its attribute *Age* has a fuzzy domain, and as consequence the class including it is fuzzy. Taking in account that students with a grade of possibility to achieve a job are really postgraduate students, the same class is fuzzy, too.

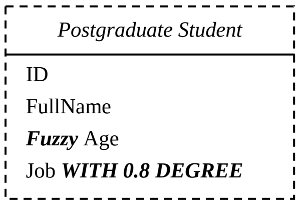


Fig. 2. Examples of a fuzzy UML class (obtained from [6])

A fuzzy association could be present in two levels of fuzziness; the first one means a fuzzy association can exist between two classes, with a grade of possibility; in the second one, an instance belongs with a membership grade to an association class. It is important to mention that with fuzzy sets, it is possible to represent "imprecise" conc-epts in a convenient form to core of a database, considering four types de values [6]:

1. Precise values (*crisp*), traditionally used in a database, but with the possibility to be included in fuzzy queries.
2. Fuzzy data represented as distributions of possibility. For example: the quality expressed as (0.4/regular, 0.7/good), where regular "<" good.
3. Attributes over "discreet, no ordered domains with analogy", with a relation of similarity. For example: hair color expressed as (1/blonde, 0.4/brown, 0.6/red).
4. Attributes defined as the last type, but without the relation of similarity.

3. Proposal of a UML Profile for a Fuzzy Multidimensional Data Model

The explanation of the profile will be made in an inverse order with respect the structure developed to the MDC that is *Attributes, Classes and Packages*. This way it begins with the smaller elements and finishes with the greater ones, which are composed of the first.

In addition to the previously mentioned, there is a fourth element, an *AssociationClass*, which could be of the *DegeneratedFact* type. This element is manipulated as a relationship between the fact table of the star schema and some dimension. Note that this is because there is an association justifying it, but it is not the association itself.

3.1 Attributes

There are five types of attributes: *Degenerate Dimension*, *FactAttribute*, *OID*, *Descriptor* and *DimensionAttribute*, which can only be present in the classes *Fact, Base* and *DegenerateFact*. The possibility to include them in the profile is indicated below.

- ***OID (OID):*** corresponds to the identifier of the *Base* classes. This element is necessary to a process of automatic export in OLAP tools, because these store it in the metadata. For this reason, it was not considered as a fuzzy element.
- ***Descriptor* (D)**: attribute derivable from another attributes, and it can be quantitative or qualitative. Without taking this into account, it can be any fuzzy type.
- ***DimensionAttribute* (DA):** gives descriptive information about an instance of *Dimension*. It can be optional and doesn't need to be specified in every element (if it is necessary, it can contain the "null" value). For its role in a dimension, it could be any fuzzy type.
- ***DegenerateDimension* (DD)**: can be considered as a dimension included in the composition of the fact table. For example, the attribute "Address" as geographical feature in a fact table, could be disaggregated into sector, street, and so on. Its presence can not be fuzzy, but its value could. For the same example, the DD *Address* could be type 3 to establish relationships of proximity between the cities and permit fuzzy queries like "*close to…*" or "*far away from…*".
- ***FactAttribute* (FA)**: attribute of the *Fact* o *DegenerateFact* classes. Consider the case of an attribute *city*; taken into account as FA not as DA, could be classified as type 1, since it is a precise data but can be used in fuzzy queries. However, if the case is such that the borders of the cities are not well defined but it is required to register such data for a house that is "within the city limits", it could be necessary to specify with what degree it belongs to that city and what degree to another. If this second situation were true, the attribute city would become type 3, which could be represented through distributions of possibility.

.

As it has been indicated already, the absence of some of the attributes that support fuzziness (*DD*, *FA*, *DA* and *D*) would repel in the data model (cube), with a degree between [0,1] in the requirements satisfied by it. For example, suppose a cube that includes the *Customer* dimension, in which the author wants incorporate fuzziness in the property of the *FullName* attribute of the *Base* class. Consider a case in which a given attribute is not present in that *Base*; transitively then, it will not be present in the *Dimension*, either. Then, to respond to queries about sales, purchases or credit notes (any be the domain) of a person, this requirement could not be solved given the absence of essential data for the problem.

The above is explained in terms of having fuzzy attributes, not fuzzy values -- since the classes cannot own values but rather attributes -- nor in terms of its presence in the model, because it has been decided to consider that aspect as fixed. In addition, the absence of the attribute can be replaced by the value "null" in this attribute.

3.2 Classes

The elements of the Class type belonging to the profile MDC are *Fact, Dimension* and *Base*. In these cases it was considered that evaluation about the presence of fuzziness must be focused on the nature of the attributes that compose them. Like in the previous section, the presentation will be made beginning with the parts -- *Base,* which through inheritance form the *Dimension,* and these with the *Fact* composition structure.

- ***Base*****:** it is a level of a hierarchy of a classification, by means of which the *Dimension* classes are defined. As an example, a *Customer* Dimension can have several Bases; one of these called *Region* Base, which owns the attribute *Location,* which can be considered of type 3 when incorporating the degree of proximity with other regions. So, that Base can be considered fuzzy while it has those fuzzy attributes.
  It is possible to consider to any *Base* as fuzzy based on the roll that their attributes play in the queries. For example, it can be more advisable to define type 3 to the bases relative to the *Dealership* dimension (concessionary), to analyze the sales grouped by region, that of the dealership not of the clients, where these last ones are associated to the dealership where they bought the vehicle.
- ***Dimension*****:** by itself does not own attributes, but an instance of the *Base* class does. In addition, only one *Base* is associated to the instance of *Dimension*, therefore it will be considered as a fuzzy *Dimension* to those associated with an instance of the class Fuzzy *Base.*
- ***Fact*****:** composed by *FactAttributes,* owned by the Fact class, and by *DegenerateDimension* attributes. Both support fuzzy data so that this class could be considered fuzzy in the way that some of their attributes are.

3.3 Packages

- ***DimensionPackage***: can contain instances of the *Dimension* and *Base* classes, where the number of dimensions is 1; this is considered fuzzy if the dimension containing it is fuzzy, meaning *Base* classes of the package can be fuzzy but not necessarily will the last one be. This choice is to avoid all of the instances of the data model being fuzzy.
- ***FactPackage*****:** can only store instances of the *Fact, Dimension or Base* classes; this package is fuzzy if the only instance of the *Fact* class is also fuzzy.
- ***StarPackage*****:** can only contain instances of the *FactPackages* or *DimensionPackages* classes. It will be considered fuzzy if the unique *FactPackage* is fuzzy as well.

3.4 Association Classes

The *DegeneratedFact* is the one element belonging to this category. The attributes can be of *DegenerateDimension* or *FactAttribute* classes. Then, it is considered if any of its attributes can contain a fuzzy value of some type.

3.5 Summary of the Profile

Considering the above, the identified elements must be associated with the definition of the metamodel of level 3, defined previously by the base profile. Figure 3 shows the relationships between the classic elements of a multidimensional data model, the elements incorporated by MDC and the *fuzzy* elements added by the present work.
To avoid ambiguous use of the profile's elements, OCL Constraints has been incorporated; as example, all of the attributes of a *FuzzyBase* must be *OID, Descriptor, FuzzyDescriptor, DimensionAttribute* or *FuzzyDimension Attribute*, which OCL expression is:

```
self.feature -> select(fe | fe.ocllsKindOf(Attribute)) -> forAll(f | f.ocllsTypeOf(OID)
or f.ocllsTypeOf(Descriptor) or f.ocllsTypeOf(FuzzyDescriptor) or
f.ocllsTypeOf(DimensionAttribute) or f.ocllsTypeOf(FuzzyDimensionAttribute))
```

where *self* references to the instance of the *FuzzyBase* class that is being evaluated.

4. Application of the Proposal in a Practice Case

To demonstrate the use of the profile, it is applied to a problem based in a library, where an analysis of information about the material lent to its different types of users is required. The dimensions of interest are:

1. Material: book, magazine, newspaper.
2. Librarian: staff responsible for giving the service.
3. User: the person who gets a material from the library.
4. Time.

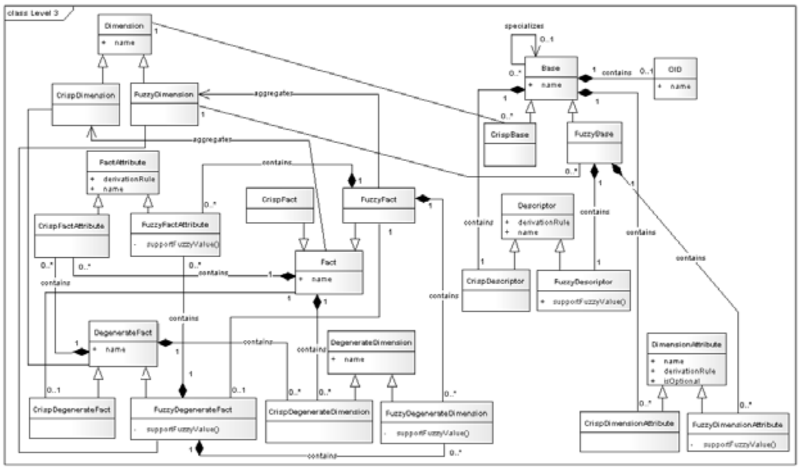


Fig. 3. Level 3 of the metamodel of the proposed profile

A fact table will register the daily loans, associated with the previous dimensions as figures 4 and 5 show. In the first, the definition of level 2 of the *Material* dimension is given. Note that fuzzy class *Material* is superclass of the other three, passing on the fuzziness. Moreover, the base class *Magazine* is fuzzy too, because its attribute *tendency* could be of *vogue*, *entertainment*, *sport*, and so on, in different grades and without an order between them (type 4).
Figure 4b gives the level 2 of the *Librarian* dimension. The base classes have been considered as fuzzy because the descriptors, which mean the type of position the person has and the function they fulfill can be imprecise or overlap.

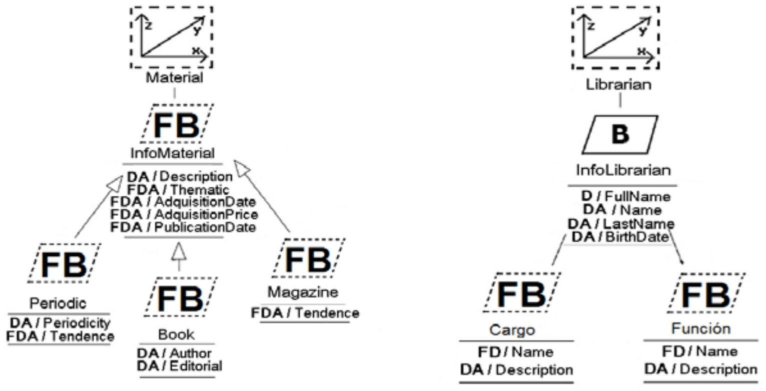


Fig. 4. Level 2 of the metamodel of the proposed profile, by the Material (a) and Librarian (b) dimensions

In the case of figure 5, the *User* dimension has been modeled as fuzzy, because it has a type 3 attribute, which is the user type (*Student, Particular, habitualUser, SporadicUser, ..*).

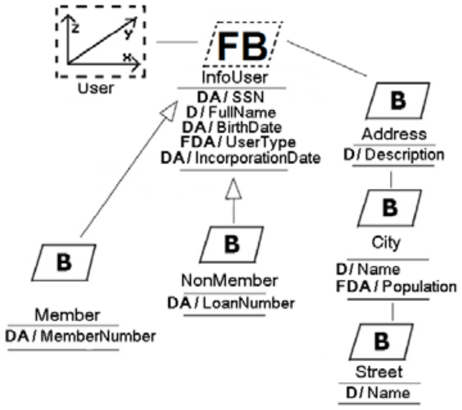


Fig. 5. Level 2 of the metamodel of the proposed profile

Figure 6 shows the model of level 2 to represent the *Fact* instance with its Dimensions associated. Considering the dimensions *User*, *Material* and *Librarian* are fuzzies, and having attributes that can be used in fuzzy queries, the *Loan* instance of *Fact* is fuzzy too.
The degenerate fact *DateBT* represents the loans start-date and length. This can occur in an association of the type "many-to-many". However, its type is fuzzy because such a date can be used in fuzzy queries, for example to list the loans out for a long time.

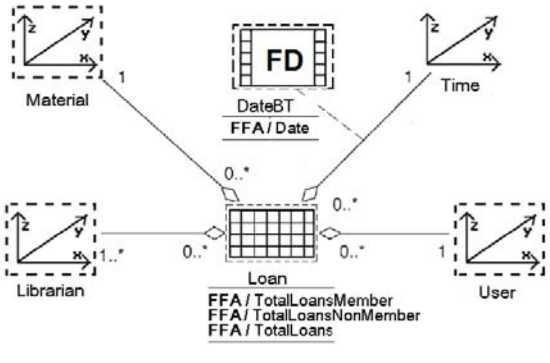


Fig. 6. Example of the level 2 of the *Material* Dimension

5. Conclusions

The research of the fuzzy logic and its inclusion in the databases, provide the first ideas to formulate a way to get a multidimensional data model with imprecise elements. This task was facilitated by a previous profile, which just contained crisp data in its structure.
Current work is concentrated on adding more precise OCL specification to the elements, to guide towards an automatic process on a CASE tool.

References

1. Molina, C., Rodríguez-Ariza, L., Sánchez D., Vila M. A. (2006). *A New Fuzzy Multidimensional Model*. IEEE Transactions on Fuzzy Systems, vol. 14(6), 897-912.
2. Object Management Group (OMG). *Unified Modeling Language Specification 1.5*. http://www.uml.org/.
3. Ludovic, A., Courtiat, J.-P., Lohr, C., Saqui-Sannes, P. de. (2004). *TURTLE: A Real-Time UML Profile Supported by a Formal Validation Toolkit*. IEEE Transactions on Software Engineering, vol. 30(7), 473-487.
4. Catalog of UML Profile Specifications. http://www.omg.org/technology /documents/profile_catalog.htm.
5. Lujan-Mora, S., Trujillo, J., Il-Yeong, S. (2006). *A UML Profile for Multidimensional Modeling in Data Warehouses*. ACM Data and Knowledge Engineering, vol. 59(3), 725-769.
6. Galindo, J., Urrutia, A., M. Piattini, M. (2006). *Fuzzy Databases: Modeling, Design and Implementation*. Idea Group Publishing.
7. Ma. Z. (2005). *Fuzzy Database Modeling with XML*. Springer.

Dynamic Spatial Approximation Trees with Clusters for Secondary Memory

LUIS BRITOS, MARCELA PRINTISTA AND NORA REYES

Dpto. de Informática, Universidad Nacional de San Luis,
Ejército de los Andes 950, San Luis, Argentina.
{lebritos, mprinti, nreyes}@unsl.edu.ar.

Abstract. *Metric space searching is an emerging technique to address the problem of efficient similarity searching in many applications, including multimedia databases and other repositories handling complex objects. Although promising, the metric space approach is still immature in several aspects that are well established in traditional databases. In particular, most indexing schemes are not dynamic. From the few dynamic indexes, even fewer work well in secondary memory. That is, most of them need the index in main memory in order to operate efficiently. In this paper we introduce a secondary-memory version of the Dynamic Spatial Approximation Tree with Clusters (DSACL-tree) which has shown to be competitive in main memory. The resulting index handles well the secondary memory scenario and is competitive with the state of the art. The resulting index is a much more practical data structure that can be useful in a wide range of database application.*

Keywords. *Similarity Search, Multimedia Retrieval, Metric Spaces.*

1. Introduction

As the growth of digital data accelerates in variety and extend, the contemporary databases are bulkier and more complex in nature. To manage this bulk and complexity increasing new techniques are employed, with the multimedia data for example, the standard approach is to search not at the level of the actual multimedia objects, but rather using characteristics extracted from these objects. In such environments, an exact match has little meaning, a very useful search paradigm is to quantify the proximity, similarity, or dissimilarity of a query object versus the objects stored in a database to be searched. Similarity or proximity searching have became a fundamental computational tasks with application in many areas as non-traditional databases, data mining, machine learning, data compression; and so on. A useful abstraction for nearness is provided by the mathematical notion of *metric space*.
In a metric space, there is a universe U of objects and a nonnegative function $d: U \times U \rightarrow \mathbb{R}^+$ defined among them that will denote a measure of "*distance*" between objects. This distance function satisfies the three axioms that make

(U, d) a metric space: strict positiveness ($d(x; y) \geq 0$ and $d(x, y) = 0 \Leftrightarrow x = y$), *symmetry* ($(d(x,y) = d(y,x))$), and *triangle inequality* ($(d(x,z) \leq d(x,y) + d(y,z))$). The smaller the distance between two objects, the more "*similar*" they are. A finite subset $X \subseteq U$ with size $n = |X|$, is called *database* and represents the collection of objects. We are interested to answer *similarity queries* posed to this database. That is, given a new object from the universe (a query) $q \in U$, we must retrieve all the elements similar enough to the query in the database. The database is preprocessed so as to build an index that reduces query time. There are two typical queries of this kind:

> **Range query**: Retrieve all elements within distance r to q in S. This is, the set $\{x \in S, d(x,q) \leq r\}$.
>
> **Nearest neighbor query** (***k*-NN**): Retrieve the k closest elements to $q \in S$. That is, a set $A \subseteq S$: $|A| = k$ and $\forall x \in A; y \in S - A; d(x,q) \leq d(y,q)$.

In this paper we are devoted to range queries. Nearest neighbor queries can be rewritten as range queries in an optimal way [6], so we can restrict our attention to range queries. In order to answer queries efficiently the database is preprocessed so as to build an *index* that reduces query time. This metric space approach to similarity search is becoming widely popular [10, 11] and a large number of indexing methods have flourished [4, 7, 10], but mature solutions from the database viewpoint are a long way off. As the topological properties of metric spaces cannot accelerate searches, which are the main aim of this work, they are not discussed here.
Most of the existing indexes are *static*: Once they are built for a given dataset, adding more elements, or removing an element from it, requires an expensive updating of the index. Some indexes tolerate insertions in principle, but their quality degrades and requires periodic rebuilding. Others tolerate deletions with the same quality degradation problem. Thus there are very few *dynamic* indexes.
There are also many interesting databases for similarity searching where the objects are so large that they must stay on disk; or the objects are so many that the index itself cannot fit in main memory. In this case, although the similarity computation can be expensive (e.g., taking milliseconds of CPU time) we cannot disregard disk costs.
From the few dynamic indexes, even fewer work well in secondary memory. That is, most of them need the data structure in main memory in order to operate efficiently. Although for some applications a static scheme may be acceptable, many relevant ones do require dynamic capabilities. Actually, in many cases it is sufficient to support insertions, such as in digital libraries and archival systems, versioned and historical databases, and several other scenarios where objects are never updated or deleted.
In this paper we introduce a dynamic index aimed at secondary memory. We base our work on the Dynamic Spatial Approximation Tree with Clusters (*DSACL-tree*) [2, 3]. It has been shown that the *DSACL-tree* gives an attractive tradeoff between memory usage, construction time, and search performance. Our secondary memory version (*DSACL*-tree*) retains these good features, and in addition perform well in secondary memory. We focus

on handling insertions and searches in this paper, leaving deletions for future work.

2. Previous Works

Algorithms to search in general metric spaces can be divided into two large areas: *pivot-based* algorithms and *compact partition-based* ones. Pivot-based algorithms are better suited for low dimensional metric spaces, while compact partitions ones deal better with high dimensional metric spaces. Although the former can improve by using more memory, they need more and more memory to beat the latter as dimension grows. On the other hand, indices based on compact partitions use a fixed amount of memory and cannot be improved by giving them more space. However, there are algorithms that combine ideas from both areas. See [10, 11, 4, 7] for more complete surveys.

Pivot-Based Algorithms The idea is to use a set of k distinguished elements (“pivots") $p_1 \dots p_k \in S$ and storing, for each dataset element x, its distance to the k pivots $(d(x,p_1) \dots d(x,p_k))$. Given the query q, its distance to the k pivots is computed $(d(q,p_1) \dots d(q,p_k))$. Now, if for some pivot p_i it holds that $|d(q, p_i)-d(x, p_i)| > r$, then we know by the triangle inequality that $d(q, x) > r$ and therefore do not need to explicitly evaluate $d(x,p)$. All the other elements that cannot be discarded using this rule are directly compared with the query.

Clustering Algorithms This second trend consists of dividing the space into zones as compact as possible, and storing a representative point (“center”) for each zone plus a few extra data that permit quickly discarding the zone at query time. Two criteria can be used to delimit a zone. The first one is the *Voronoi region*, where we select a set of centers and put every other point inside the zone of its closest center. The regions are bounded by hyperplanes and the zones are analogous to Voronoi regions in vector spaces. Let $\{c_1 \dots c_m\}$ be the set of centers. At query time we evaluate $(d(q,c_1), \dots, d(q,c_m))$, choose the closest center c and discard every zone whose center c_i satisfies $d(q,c_i) > d(q,c) + 2r$. The second criterion is the *covering radius* $cr(c_i)$, which is the maximum distance between ci and an element in its zone. If $d(q,c_i) - r > cr(c_i)$, then there is no need to consider zone i. The two criteria can be combined.

Combining Clustering with Pivots There are some indexes that combine both ideas by dividing the space into compact zones and, at the same time, storing distances to some distinguished elements (pivots) [1].

3. Dynamic Spatial Approximation Trees

In this section we will describe briefly the *Dynamic Spatial* Approximation Tree (*DSA-tree*), in particular the version called *timestamp with bounded arity* (reported in [8] as one of the better options for this dynamic tree), on top of which *DSACL-tree* [2, 3] was built. The *DSA-tree* is a data structure to answer similarity queries in metric spaces based on the concept to approach the query spatially, getting closer and closer to it, so when we look for an element from the universe (a query $q \in U$) and being in some element a belonging to the database S ($S \subseteq U$), the goal is to move to another object of S spatially closer of q than a. When is not longer possible do this move anymore, we are positioned on the element closest to q from S.
The *DSA-tree* is built incrementally via insertions. The tree has a maximum arity A. Each tree node a stores a *timestamp* of its insertion time, *time(a)*, its *covering radius, R(a)*, and its set of children *N(a)* (the *neighbors of a*). To insert a new element x, its point of insertion is sought starting at the tree root and moving to the neighbor closest to x, updating *R(a)* in the way. We finally insert x as a new (leaf) child of a if (1) x is closer to a than to any $b \in N(a)$, and (2) the arity of a, $|N(a)|$, is not already maximal. In other case, we insert x in the subtree of the closest $b \in N(a)$. Neighbors are stored left to right in increasing timestamp order, and each element is older than its children and its next sibling.
The idea for range searching is to replicate the insertion process of relevant elements. That is, we act as if we wanted to insert q but keep in mind that relevant elements may be at distance up to r from q. So in each decision for simulating the insertion of q we permit a tolerance of $\pm\, r$, so that it may be that relevant elements were inserted in different children of the current node, and backtracking is necessary.
We have to consider two facts, at the time an element x was inserted. The first that, a node a in its path may not have been chosen as its parent because its arity was already maximal. So, at query time, instead of choosing the closest to x among $\{a\} \cup N(a)$, we may have chosen only among $N(a)$. Hence, we perform the minimization only among elements in $N(a)$. The second fact is that, elements with higher timestamp were not yet present in the tree, so x could choose its closest neighbor only among elements older than itself.
Hence, we consider the neighbors $\{b_1,\ldots,b_k\}$ of a from oldest to newest, disregarding a, and perform the minimization as we traverse the list. This means that we enter into the subtree of b_i if $d(q,b_i) \leq \min\{d(q,b_1),\ldots,d(q,b_{i-1})\}+2r$. Up to now we do not really need the exact timestamps but just to keep the neighbors sorted by timestamp. We make better use of the timestamp information in order to reduce the work done inside older neighbors. Say that $d(q, b_i) > d(q, b_{i+j})+2r$. We have to enter into the subtree of b_i anyway because b_i is older. However, only the elements with timestamp smaller than that of b_{i+j} should be considered when searching inside b_i; younger elements have seen b_{i+j} and they cannot be interesting for the search if they are inside b_i. As parent nodes are older than their descendants, as soon

as we find a node inside the subtree of b_i with timestamp larger than that of b_{i+j} we can stop the search in that branch, because its entire subtree is even younger.

4. Spatial Approximation between Clusters

In this section we will describe briefly the *Dynamic Spatial Approximation Trees between Clusters* (*DSACL-tree*) [2, 3]. The *DSACL-tree* performs the spatial approximation on groups or *clusters* of objects that are very close to each other, rather than individual objects. By this way it can reduce search costs, because it has to do less backtracking. Therefore, in the *DSACL-tree* each node represents a cluster of very similar objects, for short we refer to it simply as *cluster*. Thus, we relate the clusters by their proximity in the metric space. So, each node of the tree would be able to store multiple database objects, reducing the number of nodes with respect to the original *DSA-tree*.

As in the *DSA-tree* we build the tree incrementally, considering a *maximum arity* and maintaining information of the *timestamp* (time of insertion of each element). We also register the timestamp *time(c)* of each node c in the tree, which is the time when this node was created. Each node *c* keeps an object *center(c)* as the *center* of the cluster and the *k nearest objects* (*cluster(c)*) seen in its subtree, and is connected with their *clusters-neighbors N(c).* The cluster also have a cluster radius *rc(c),* that is considering the objects in increasing order to the *center(c)* the distance of the *k*-th object in the *cluster(c).* Any object further away from the center than *rc(c)* would become part of another tree node, which could be a new neighbor in some cases, since the arity is bounded in the same way as *DSA-tree*. Each node c also stores the maximum distance between the *center(c)* and the farthest object in its subtree *R(c)* (as *DSA-tree* does), called *covering radius* of the subtree of *c*.

Since each node *c* represents a cluster centered in *center(c)* with at most *k* objects within *cluster(c),* we maintain the distances between *center(c)* and all the objects in *cluster(c)* ordered by increasing distance to the center. At search time, we can use these stored distances in order to minimize the number of distance computations using the triangle inequality. Besides, if $x_1,\ldots, x_k$ are the objects in *cluster(c)* sorted by distances, the covering radius of the cluster will be $rc(c) = d(center(c),x_k)$. Therefore, for each object x_i inside the cluster, we stored its insertion moment $time(x_i)$ and the distance $d(center(c), x_i)$. It is clear that it is not necessary to really register *rc(c)* because it can be obtained from the stored distances inside the node. During searches, both radii *rc(c)* and *R(c)* are used to rule out entire areas of space containing non relevant elements.

Because of the spatial approximation, to insert a new element *x*, we should go down the tree until found the node *c* such that *x* is closer to *center(c)* than the centers of neighbors in *N(c).* If in *cluster(c)* there is room for one more element, then it will be inserted along with its distance. If there is not room, we must choose the most distant element *b* among the *k* elements in *cluster(c)* and *x* (*k* + 1-th in distance order from *center*(*c*)). We have two possible cases:

1. if *b* is *x*, then *x* must be added like center of a new neighbor node of *c*, if the arity allows it, otherwise it must choose the node among all the neighbors in *N(c)* whose center is the nearest and keep the insertion from there.
2. if *b* is not *x*, then *b* must choose the nearest center a among *center(c)* and the center of all nodes neighbors in *N(c)* that are newer than *b* because when *b* was inserted, it was not compared with them. Later, if *a* is *center(c)*, the process followed is the same as when *b* is *x;* otherwise, if *a* is not *center(c)*, then continues with the insertion of *b* from the node with center *a*.

Algorithm 1 illustrates the whole insertion process. The function is invoked as `InsertCl`(*a, x*), where *a* is the root node and *x* is the element to be inserted.

When performing a range query, we proceed in a similar way as *DSA-tree*, which is we perform the spatial approximation to the query via the centers of nodes. As we mentioned previously, the idea for range searching is to replicate the insertion process of the relevant elements to the query. That is, we act as if we wanted to insert *q* but keeping in mind that relevant elements may be at distance up to *r* from *q*. So that it may be that relevant elements were inserted in a cluster, in different children of the current node, and backtracking is necessary. The range search process is shown in Algorithm 2.

Algorithm 1: Insertion algorithm in a *DSACL-tree* with root node *a*

```
InsertCl (Node $a$, Element $x$)
1.  $R(a) \leftarrow$ max$(R(a), d(center(a), x))$
2.  If $((|cluster(a)| < k) \vee (d(center(a), x) < rc(a)))$ Then
3.     $cluster(a) \leftarrow cluster(a) \cup \{x\}$, $d'(x) \leftarrow d(center(a), x)$
4.     $timestamp(x) \leftarrow CurrentTime$
5.     If $(|cluster(a)| = k + 1)$ Then
6.        $y \leftarrow argmax_{z \in cluster(a)}\ d'(z)$
7.        $cluster(a) \leftarrow cluster(a) - \{y\}$
8.        InsertCl$(a,y)$
9.  Else
10.    $c \leftarrow argmin_{b \in N(a)}\ d(center(b), x)$
11.    If $((d(center(a), x) < d(center(c), x)) \wedge (|N(a)| < MaxArity))$
        Then /* $b$ is a new node, neighbor of $a$, with $center(b) = x$ */
12.       $N(a) \leftarrow N(a) \cup \{b\}$
13.       $center(b) \leftarrow x$, $cluster(b) \leftarrow \phi$, $N(b) \leftarrow \phi$, $R(b) \leftarrow 0$
14.       $timestamp(x) \leftarrow CurrentTime$, $time(b) \leftarrow CurrentTime$
15.    Else
16.       InsertCl $(c,x)$
```

5. Secondary Memory

The distance is assumed to be expensive to compute. However, when we work in secondary memory, the complexity of the search must consider both the number of distance evaluations performed and the I/O time; other components such as CPU time for side computations can usually be disregarded. Given a dataset of $|S| = n$ objects of total size N and disk page size B, queries can be trivially answered by performing n distance evaluations and N/B I/Os. The goal of an index is to preprocess the dataset so as to answer queries with as few distance evaluations and I/O operations as possible.

Algorithm 2: Range query algorithm on a *DSACL-tree* with root node *a*

```
RangeSearchCl (Node $a$, Query $q$, Radius $r$, Timestamp $t$)
1.  If $((time(a) < t) \wedge (d(center(a), q) \leq R(a) + r))$ Then
2.     If $(d(center(a), q) \leq r)$ Then Report $a$
3.     If $((d(center(a), q) - r \leq rc(a)) \vee (d(center(a), q) + r \leq rc(a)))$ Then
4.       For $c_i \in cluster(a)$ Do
5.         If $|(d(center(a), q) - d'(c_i)| \leq r$ Then
6.           If $d(ci, q) \leq r$ Then Report $c_i$
7.       If $(d(center(a), q) + r < rc(a))$ Then Return
8.     $d_{min} \leftarrow \infty$
9.     For $b_i \in N(a)$ in increasing order of timestamp Do
10.      If $(d(center(b_i), q) \leq d_{min} + 2r)$ Then
11.        $k \leftarrow min\{j > I, d(center(b_i), q) > d(center(b_j), q) + 2r\}$
12.        RangeSearchCl $(b_i,q,r,time(b_k))$
13.      $d_{min} \leftarrow \min\{d_{min}, d(center(b_i), q)\}$
```

The *DSACL*-tree* (*DSACL-tree* in secondary memory) make also a partition of the searching space considering spatial proximity, grouping the closest elements, relating complete clusters by its proximity in the space. This permits that each node of the structure is capable of storing multiple elements from the database. Because of each node has a fixed size, this structure seems to be naturally adequate for secondary memory. To avoid disk underutilization, *DSACL*-tree* we will *x* the number of size of the clusters and also the maximum arity of the tree in function to the available page size. Therefore, each node takes exactly one page in disk, simplifying the administration of nodes. Therefore, the *DSACL*-tree* is an improvement of the *DSACL-tree* because it maintains the same structure of the index, while also works efficiently in secondary memory.
As in the original *DSACL-tree* does, for each neighbor of a node *a*, we will save its object center, its location on the file and its insertion time. We do this to avoid, as we shall see, some I/Os on insertions. Because we need to set the size of a node as a size of a page disk, considering the size needed to represent an element we must to *x* the maximum arity and the cluster size of each node. If the elements are big it is possible to notice that the arity and the

cluster size will be small. However, as it has been demonstrated in [8], it is not a drawback because small arities were a key factor, for the *DSA-tree*, to reduce construction and search costs in several metric spaces.
To insert an element *x* into our structure we proceed exactly like in the *DSACL-tree*: We find the insertion point in the tree, following a unique path, so that when we determine that *x* should be added to a node a because *x* is closer to a than to any neighbor in *N(a)*, already have loaded the page corresponding to a. If a already have its k elements then it must choose the element furthest from the center from its $k+1$ elements and then choose if *x* must to be inserted like center of a new neighbor, if the arity allows it, otherwise the insertion must to continue forcing *x* to choose the closest neighbor from *N(a)* and keeping going down on the tree recursively.
As mentioned earlier, in every node (page) we store the object center of all its neighbors, this avoid some I/O operations when the element to insert must to decide which element is closer the center of the node or some neighbor.

6. Experimental Results

In order to give a broad picture of the performance of our index, we have selected four widely different metric spaces, all from the SISAP Metric Space Library (www.sisap.org). The metric spaces considered are:

– WORDS: a dictionary of 69,069 English words. The distance is the edit distance, that is, the minimum number of character insertions, deletions and substitutions needed to make two strings equal.

– DOCUMENTS: 1,265 documents under the Cosine similarity, from TREC-3 collection. In this model the space has one coordinate per term and documents are seen as vectors in this space. The distance we use is the angle among the vectors.

– IMAGES: 40,700 20-dimensional feature vectors, generated from NASA images, using Euclidean distance.

– HISTOGRAMS: 112,682 8-D color histograms (112-dimensional vectors) from an image database. Euclidean distance is used.

For search experiments, we built the indexes with 90% of the objects and used the other 10% (randomly chosen) as queries. All our results are averaged over index constructions using different database permutations. We have considered range queries retrieving on average 0.01%, 0.1% and 1% of the dataset. This corresponds to radii 0.14, 0.15 and 0.195 for DOCUMENTS, 0.60574, 0.78 and 1.009 for IMAGES, and 0.051768, 0.082514 and 0.131163 for HISTOGRAMS.
WORDS have a discrete distance, so we used radii 1 to 4. The same queries were used for all the experiments on the same datasets.
In [2, 3] it was experimentally demonstrated that *DSACL-tree* can beat *DSA-tree* in some of these metric spaces. So, we only show here the behavior of

DSACL-tree* for lack of space. As it can be noticed in Fig. 1, the maximum arity has a tradeoff with the cluster size, and this tradeoff affects the number of I/O operations performed. If the arity is small, the cluster can increase its size, and it has showed to be good to minimize the I/O operations (see Fig. 2). In WORDS the better results is with maximum arity of 2, considering distance evaluations and I/O operations performed during searches. For IMAGES and DOCUMENTS the maximum arity would be 4. In the case of HISTOGRAMS, because the size of each element (112 real numbers are needed to represent each element), the maximum arity allowed is 2.
It is also important to notice that our secondary memory version of the *DSACL-tree* have a good fill ratio, in all cases over 66%. Table 1 shows the average disk page occupancy achieved for the different spaces. In [9] are presented two versions for secondary memory for the *DSA-tree* (*DSA*-tree* and *DSA+-tree*), and the experimental results for them and for the M-tree [5], on the same four metric spaces. We compare the fill ratio and the total number of pages used by these data structures with our results. As it can be noticed we obtain a good fill ratio and we use, in general, fewer disk pages than the other indices designed for secondary memory, while we maintain a good search performance.

Table 1. Average space usage for the different datasets

	Fill ratio			**Total pages used**			
Dataset	***DSA*-tree***	***DSA+-tree***	***DSACL*-tree***	***DSA*-tree***	***DSA+-tree***	***M-tree***	***DSACL*-tree***
WORDS	83%	66%	69%	904	1,536	1,608	885
DOCUMENTS	84%	68%	68%	12	22	31	9
IMAGES	80%	67%	72%	1,271	1,726	1,973	1,310
HISTOGRAMS	75%	67%	66%	18,781	21,136	31,791	18,827

7. Conclusions

In this work we present the *DSACL*-tree* which is an index for searching metric spaces for secondary memory. This new index maintains the good features of the *DSACL-tree* (spatial approximation, dynamism, and clustering), but also takes into account the I/O operations costs. In fact, each node of the *DSACL*-tree* corresponds to a page. By this way, we try to get the most advantage in each read or write into the disk, locating similar objects together. Therefore, we reduce the backtracking at searches improving the cost, in distance evaluations, at the same time we make few I/O operations during the retrieval relevant elements. We have shown some empirical evidence that our new index is competitive considering the space used, with respect to the other dynamic indices for secondary memory such as *DSA*-tree*, *DSA*-+tree* and *M-tree*.
For the final version of this paper we plan to include the complete comparison *of DSACL*-tree* with the other secondary memory indices.

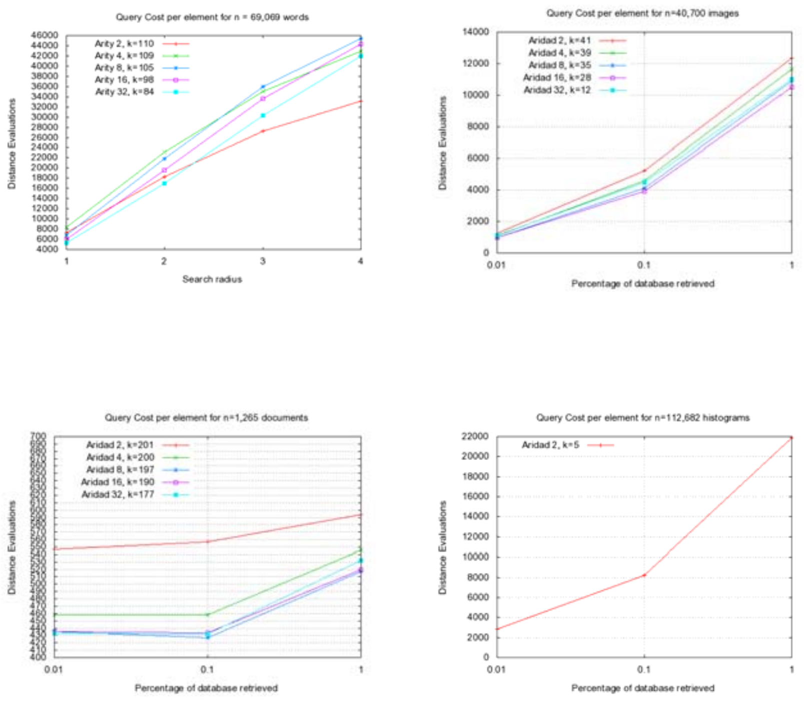


Fig. 1: Distance evaluations at search time, for the four spaces

References

1. Arroyuelo, D., Muñoz, F., Navarro, G., Reyes, N. (2003). Memory-adaptative dynamic spatial approximation trees. In: Proc. 10th International Symposium on String Processing and Information Retrieval (SPIRE). 360-368. LNCS 2857, Springer.
2. Barroso, M., Navarro, G., Reyes, N. (2005). Combinando clustering con aproximación espacial para búsquedas en espacios métricos. In: Actas del XI Congreso Argentino de Ciencias de la Computación (CACIC). Concordia, Argentina, in Spanish.
3. Barroso, M., Reyes, N., Paredes, R. (2010). Enlarging nodes to improve spatial approximation trees. In: Proc. 3rd International Workshop on Similarity Search and Applications (SISAP). 41-48. ACM Press.
4. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J. (2001). Searching in metric spaces. ACM Computing Surveys 33(3), 273-321, Sep.
5. Ciaccia, P., Patella, M., Zezula, P. (1997). M-tree: an efficient access method for similarity search in metric spaces. In: Proc. of the 23rd Conference on Very Large Databases (VLDB'97). 426-435.

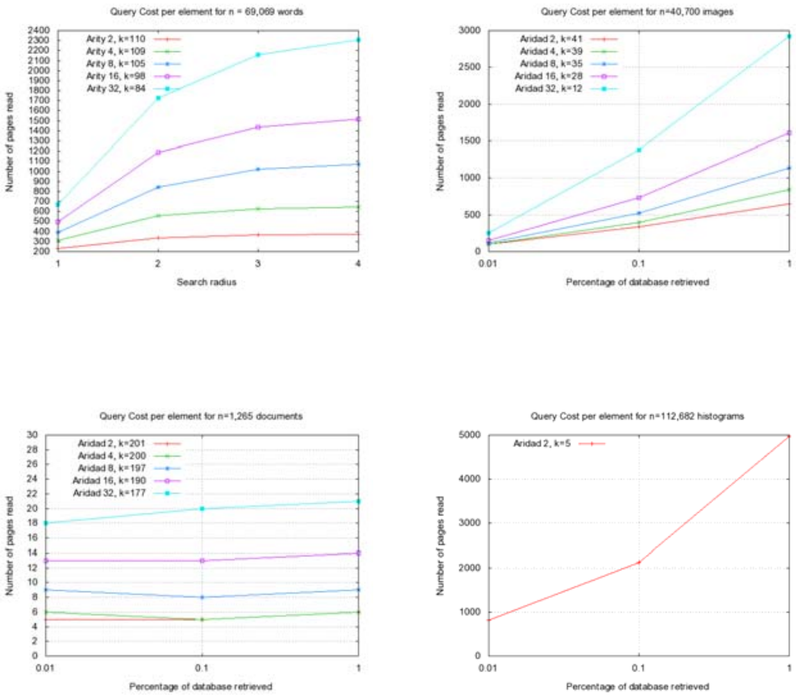


Fig. 2: Number of disk pages read at search time, for the four spaces.

6. Hjaltason, G., Samet, H. (2000). Incremental similarity search in multimedia databases. Tech. Rep. CS-TR-4199, University of Maryland, Computer Science Department.
7. Hjaltason, G., Samet, H. (2003). Index-driven similarity search in metric spaces. ACM Trans. on Database Systems 28(4), 517-580.
8. Navarro, G., Reyes, N. (2008). Dynamic spatial approximation trees. ACM Journal of Experimental Algorithmics (JEA) 12, article 1.5, 68 pages.
9. Navarro, G., Reyes, N. (2009). Dynamic spatial approximation trees for massive data. In: Proc. 2nd International Workshop on Similarity Search and Applications (SISAP). pp. 81-88. IEEE CS Press.
10. Samet, H. (2006). Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann, San Francisco, CA, USA.
11. Zezula, P., Amato, G., Dohnal, V., Batko, M. (2006). Similarity Search: The Metric Space Approach, Advances in Database Systems, vol. 32. Springer.

V

Architecture, Nets and Operating Systems Workshop

Collection and Publication of a Fixed Text Keystroke Dynamics Dataset

LUCIANO BELLO[1], MAXIMILIANO BERTACCHINI[1], CARLOS BENITEZ[1], JUAN CARLOS PIZZONI[1] AND MARCELO CIPRIANO[2]

Si6 Labs - CITEFA - Inst. de Investigaciones Científicas y Técnicas para la Defensa.
{lbello,cbenitez,mbertacchini,jpizzoni}@citefa.gov.ar.
Escuela Superior Técnica, Facultad de Ingeniería del Ejército,
Buenos Aires, Argentina.

Abstract. *Keystroke Dynamics is a powerful technique which allows to detect and identify intruders in computer systems. In order to test keystroke data pattern matching and clustering algorithms, user data collection is a mandatory task. Si6 Labs3[1] developed a web application named k-profiler4[2] with the purpose of collecting the typing rhythm data of volunteer users. This paper describes the experiment design criteria as well as the format of the collected data which will be used for Si6 projects and will be publicly available.*

1. Introduction and Previous Work

During the last 30 years many works have been published about computer user identification and/or individualization based on Keystroke Dynamics techniques. These methods are based on measuring the latency between successive keystrokes in a computer keyboard. The most common application of these techniques is the reinforcement of user authentication in computer systems based on the user keystroke pattern [2,6,7]. In the last years, these method was also translated to smartphones [9,12]. Later just a few papers have been published on intruder identification based on Keystroke Dynamics [13].
All of the abovemetioned papers use their own collected user data which make it difficult to compare results of different algorithms because of the lack of a common dataset. It can be pointed out that there is a related research area which is intruder or masquerader identification in UNIX systems based on user command line behavior [14,15,17,19]. Three datasets of UNIX user command line data were published and are used by most works in the area. These datasets were collected by Samuel Greenberg[20] and Mathias Schonlau[21], and a synthetic one created by Ramkumar Chinchani et al.[22]. In recent years some keystroke datasets have been published, such as [23][3] and

[1] http://www.citefa.gov.ar/si6/
[2] http://www.citefa.gov.ar/si6/k-profiler
[3] http://www.cs.cmu.edu/~keystroke/

[24][4]. These datasets are based on fixed short text, such as a particular password, with focus in authentication. This approach is suitable for user authentication but not for user identification. Under these conditions, the natural user typing rhythms get lost because she is not familiar with the keyboard (layout, position or size). Moreover, she types under pressure just one fixed given word, causing a deformation of her keystroke pattern along the session. In the presented dataset, each volunteer types natural sentences on her own keyboard, watching her own screen without any pressure or external disturbing factors.
The goal of this work is an attempt to provide a standard dataset to be used in future works in Keystroke Dynamics research area, particularly in user identification. This dataset will save other researchers the complex task of collecting keystroke data from different users and will allow the comparison of different algorithms.

2. The Dataset collector

The keystroke collector was designed with flexibility in mind to cover as many use cases as possible. Both depressed and released key times were recorded as the user typed 15 Spanish sentences. Since much work has been perfomed in the past in relation with profiling users in UNIX command line environments[14,15,17,19], 15 UNIX commands are showed at the last page. Volunteers were asked to type these 16 paragraphs along with some enrollment data (see Section 2.5) which were used to anonymously label them.
Nowadays, this task is performed on a regular basis, so the dataset keeps growing. At the moment, more than 66 keystroke profiles have been collected.

2.1 The Web Application

A web-based keystroke collector was developed based on PHP and JavaScript. The code was deployed at http://www.citefa.gov.ar/si6/k-profiler/. A web-based approach was chosen in order to increase the potential dataset size, since it is inherently multiplatform and widely accessible, besides the fact that the volunteer types on her own keyboard. k-profiler is capable of capturing key depress or release times using the onkeyup and onkeydown Javascript events on the client-side (i.e. the web browser). This data is sent via POST HTTP method and stored in the server.
Once the user logs in, she is asked to fill in some basic enrollment data (see Section 2.5) and later she is prompted to type the paragraphs grouped in 6 pages.

[4] http://www.cs.cmu.edu/~keystroke/
http://jdadesign.net/2010/04/pressure-sensitive-keystroke-dynamics-dataset/

The order in which paragraphs and commands are shown to the user is generated randomly, in order to prevent biased keystroke timings in the last texts due to tiredness. As a result, the user is rewarded with a 3D chart which shows the average time for each digraph.

2.2 Limitations

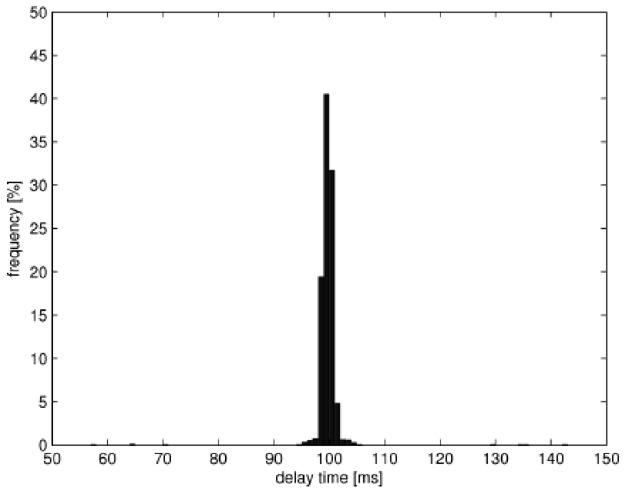


Fig. 1. Example of k-profiler time measure errors

The JavaScript code is executed at the client-side and the keystroke event time precision is strongly dependent on the conditions of the platform where the client browser is working (i.e. process priority, RAM and CPU state, browser load, etc.). The timing accuracy has been measured in different computers, operating systems and browsers by sending a key at a fixed time interval. The measured error was at about 10% with a slow CPU use and a maximum of 20% with high CPU load. Figure 1 shows a histogram sample of measured delays. In this example, keyboard repetition rate was set at 100 ms, CPU load was about 10% and as result, the error was of 12 ms.

Regarding this, the time resolution measured in JavaScript is on the order of tenths of miliseconds, while the values in experiments having special software for keystroke collection is on the order of 200 microseconds[23]. This is a system limitation, but taking into account the fact that digraph intervals usually range between 20 and 500 miliseconds, it is considered acceptable.
In browsers running on the GNU/Linux operating system, the onkeydown event for dead keys, used for acute accent (´) on Spanish vowels, can not be recorded. In those cases, an onkeyup event with keyCode 0 followed by the

onkeyup of the vowel, without a dn, is detected and stored. Under these browsing conditions, when a modifier key (e.g. Shift or AltGr) is used, the event onkeyup is detected twice sometimes. Since the web application captures events as output of the keyCode and charCode methods, and these depend on the particular browser implementation[5] , it is important to keep in mind the user's user-agent and the keyboard layout to detect particular keystrokes.

2.3 Text Selection

In this dataset, a fixed but natural text approach was choosen. This text was carefully selected to hold some statistical properties (see Section 2.4) from the following pieces of literature in the public domain:

- *One Thousand and One Nights or Arabian Nights*;
- *War and Peace*;

The first one (whose translation in Spanish is Las mil y una noches), is a compilation of ancient arabian stories; and the second one (whose translation in Spanish is Guerra y paz), is the famous Russian novel by Leon Tolstoy. Plain text editions of both books were taken and their sentences were splitted using the tokenize() function from the NLTK Python library [27].
After unifying the capitalization and purging some special chararcters (e.g. all the simple and double quotes, dashes and hyphens), all the repeted sentences and those with less than 70 characters were discarded.
The total amount of digraphs in these sentences was counted and a ranking including the ten most popular ones was created. The same work was performed on words, resulting in two lists: top digraphs and top words.
Under the assumption that these are the most popular digraphs and words in the language, two rankings were produced, sorted by the percentage of the sentences covered by the popular digraphs or words. The intersection of the top 30 of these rankings is a set of 20 sentences, from which 15 were arbitrarily selected, excluding those that included unusual syntax forms. These sentences are listed below.
ks 00 en aquel momento estaba tan seguro de ello como si se encontrase a su lado al pie del altar.
ks 01 en todas partes se hablaba de la guerra y de que el enemigo estaba a las puertas de la ciudad.
ks 02 pedro se daba cuenta de que era el centro de la atención general y se sentía contento y cohibido.
ks 03 pero le era penoso que el estado de espíritu de las personas que ten´ delante estuviera tan alejado del que nacía en ella.
ks 04 porque las paredes de la casa y las de la cuadra se han derrumbado encima de todo lo que había en la casa, sin excluir a los carneros, los gansos y las gallinas.
ks 05 se los veía en los patios y en las ventanas de las casas; otros se agrupaban en la calle.

---

[5] http://mozilla.pettay.fi/moztests/events/browser-keyCodes.htm.

ks 06 aprendí también la ciencia de los astros y las palabras de los poetas.
ks 07 y sentáronse los tres ante las bandejas de oro debidas a los cuidados del genio de la lámpara; y aladino estaba sentado en medio, con su esposa a la derecha y su madre a la izquierda.
ks 08 a causa del juego de luces entre las copas de los tilos, no podía darse cuenta del cambio de las caras.
ks 09 al darse cuenta de la presencia del príncipe, se detuvo perpleja en el umbral de la puerta.
ks 10 la condesa se dirigió a la sala de los iconos y sonia la halló arrodillada delante de las pocas cruces que todavía pendían de las paredes.
ks 11 no es que dijera aquello que pudiese complacerla, sino que juzgaba desde el punto de vista de ella todo lo que decía.
ks 12 una de ellas estaba junto a la cabeza del califa y la otra a sus pies.
ks 13 después de despedirme del rey y de todos los amigos que me hice durante mi estancia en aquella isla tan encantadora, me embarqué en la nave, que enseguida se dio a la vela.
ks 14 y efectivamente, me dio la tentación de deshacerme de aquel collar de oro y de perlas.
During the data collection, sentences are displayed to the volunteer in a random order and can be identified in the dataset by the prefix “ks ”. In the last page, 15 UNIX commands are added, also randomly sorted. These commands are the 15 most frequent commands from a combination of 3 datasets ([20], [15] and a private one obtained from Si6 Labs honeypots. They can be identified inside the dataset with the “cm “ prfix. These commands are listed below.
cmd_00 ls -a
cmd_01 bash
cmd_02 rm -rf /var/log/lastlog
cmd_03 unset HISTSAVE
cmd_04 cat /etc/passwd
cmd_05 wget
cmd_06 fg %2
cmd_07 more Makefile
cmd_08 lpq -Pip
cmd_09 tar xvfz *.tgz
cmd_10 mv a.out /var/tmp
cmd_11 touch wtmp
cmd_12 ps aux
cmd_13 kill -9 0

2.4 Statistical Features of the Fixed Text

The ten most frequent digraphs in the dataset, sorted by its repetition rate in the selected text, are show in Table 1. Most digraphs include a white space. The top ten digraphs without white spaces and their frequency are: ’de’ (57), ’la’ (52), ’en’ (38), ’as’ (33), ’ue’ (25), ’es’ (24), ’el’ (24), ’os’ (23), ’ta’ (22) and ’nt’ (22).

The ten most popular words are listed in Table 2. With 333 words in the 15 sentences, the repetition rate is calculated as 100 $*$ sum/333, where column sum is the sum of ocurrences of each digraph in each sentence. These listed words comprise nearly 44% of the total words in the text.

2.5 Enrollment Data

The initial form of the web application asks the volunteer for an e-mail (or name or nickname), occupation, age, handedness and keyboard layout. Additionally, the IP address and browser User-Agent are recorded. These two pieces of data together with the e-mail are used to detect user reentrance or whether the same person performed the experience twice. The IP address is discarded during the anonymization process (see Section 2.6) and the e-mails/names/nicknames from the same user are joined.

2.6 Anonimization and Publication

digraph	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	sum
’e ’	4	5	5	7	6	3	2	5	5	4	6	8	1	10	6	77
’a ’	2	6	5	4	8	2	2	9	4	5	9	4	9	7	1	77
’ d’	2	3	3	4	4	1	2	5	6	5	5	5	2	6	6	59
’de’	2	3	2	4	5	1	2	6	5	4	6	5	2	5	5	57
’la’	1	4	1	3	6	3	3	5	2	2	8	3	3	5	3	52
’ l’	1	3	1	2	9	5	4	7	4	2	5	1	2	3	1	50
’s ’	**0**	4	**0**	2	7	7	5	7	5	**0**	6	1	2	4	**0**	50
’ e’	3	3	2	7	3	3	**0**	3	1	2	0	3	2	6	1	39
’en’	3	2	6	3	2	4	2	4	2	3	1	**0**	**0**	4	2	38
’o ’	5	1	3	4	3	**0**	**0**	4	3	1	**0**	6	1	1	2	34

Table 1. Digraph repetition by sentence number

word	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	sum	repetition rate
’de’	1	3	2	2	3	1	2	2	3	2	3	2	1	2	4	33	9.91
’la’	0	2	1	0	3	1	1	3	0	2	3	0	2	2	1	21	6.31
’y’	0	1	2	0	2	1	1	3	0	0	1	0	1	1	2	15	4.50
’que’	0	1	1	3	1	0	0	0	0	0	1	4	0	2	0	13	3.90
’las’	0	1	1	3	1	0	0	0	0	0	1	4	0	2	0	13	3.90
’los’	0	1	1	3	1	0	0	0	0	0	1	4	0	2	0	11	3.30
’en’	1	1	0	1	1	3	0	1	0	1	0	0	0	2	0	11	3.30
’a’	1	1	1	2	2	1	0	3	2	0	2	1	2	1	0	11	3.30
’se’	1	1	2	0	1	2	0	0	0	1	1	0	0	1	0	10	3.00
’del’	1	0	0	1	0	0	0	1	2	1	0	0	1	1	0	8	2.40

Table 2. Word repetition by sentence number

The anonymized dataset is published at http://www.citefa.gov.ar/si6/k-profiler/dataset/. The anonimization process removes sessions with less than 12 completed and accepted sentences. The finished ones are tagged as finished, and as unfinished otherwise. The occupation and field of each session are normalized and translated. The e-mail/name/nickname is replaced by a generic string with the prefix ”user ”. The date, age and user-agent is kept unchanged. Each session is saved in a file the following filename: <UNIX-timestamp> <user>.[un]finished. Finally these files are archived and compressed as kprofiler-<date>-<UTCtime>.tar.gz.

3. The Published Dataset (kprofiler-20100716-1442)

The published file was relesed with the data collected up to July 16th, 2010. It
includes 66 sessions (58 finished and 8 unfinished) from 63 unique volunteers,
from whom 54 of them finished the whole process.

3.1 Data Format

A session file (see Figure 2 for an example) is named after the UNIX timestamp in which the user started and the unique volunteer id. The files are in UNIX format, ASCII encoded. The first line contains the following fields separated by semicolons: Action, local UNIX timestamp, date and hour, user id, occupation, field, age, gender, handedness, keyboard layout, obfuscated IP (normaly “na”) and user-agent.

```
;SESSION;1269898779;2010-03-29 18:39:39;user_146;Other;Art;22;Female;right-handed;\
latam;na;Mozilla/5.0 (Windows; U; Windows NT 5.1; es-AR; rv:1.9.2.2) \
Gecko/20100316 Firefox/3.6.2 (.NET CLR 3.5.30729);text/html,application/xhtml+xml,\
application/xml;q=0.9,*/*;q=0.8;es-ar,es;q=0.8,en-us;q=0.5,en;q=0.3;gzip,deflate;\
ISO-8859-1,utf-8;q=0.7,*;q=0.7

ks_14 1269898815055 dn 85
ks_14 1269898815135 up 85
ks_14 1269898815462 dn 8
(...)
ks_13 1269898881950 up 65
ks_13 1269898881982 dn 190
ks_13 1269898882054 up 190

---------PAUSED---------
2010-03-29 18:41:35
---------PAUSED---------
;RESUME;1269904718;2010-03-29 20:18:38;user_146;Other;Art;22;Female;right-handed;(...)

ks_06 1269904803245 dn 65
ks_06 1269904803332 up 65
ks_06 1269904803452 dn 80
(...)
cm_13 1269905257644 up 85
cm_13 1269905257876 dn 88
cm_13 1269905257940 up 88

---------END---------
2010-03-29 20:27:37
---------END---------
```

Fig. 2. Session example

The Action can be SESSION or RESUME. In the first case, the user started a new session. If the user is resuming an existent session, RESUME will appear and the date will indicate when this happened. If the user paused the data collection in order to continue later, a banner saying -----PAUSED---- and the date and hour are shown.

Keystroke data follows line by line in the format [invalid-]<model id> <UNIX timestamp in miliseconds> <direction> <char code>. The invalid tag appears when the detected amount of keystrokes is out of bounds (when they are more than 1.3 or less than 0.95 times the expected amount of keys). In those cases, the sentence or the command is showed again to the volunteer and she is asked to retype it. The invalid keystrokes are surrounded between the tokens invalid-data-start and invalid-data-end.
Instead of comparing the typed data against the original data, keystroke counts were used as aprove/reject parameter. So, if the user typed less than 90% or more than 130% of the total sentence characters, the sentence will be rejected and she will be asked to type it again. Mistakes were allowed so that data on typing errors can also be used for user identification [?].
The <model id> refers to the sentence or the command that the volunteer typed. The Event may be dn or up, depending whether the key was depressed or released respectively. The char code is the output of the following JavaScript command:
e.keyCode? e.keyCode : e.charCode;

where e is the keystroke event. This returns the output of keyCode[6] when this function is implemented for the event or charCode otherwise, which is the ASCII value of the resulting character. In browsers running on GNU/Linux, the dn of a key associated with a dead key (and the deadkey itself) is not detected and only the up is shown (see Section 2.2 for details).

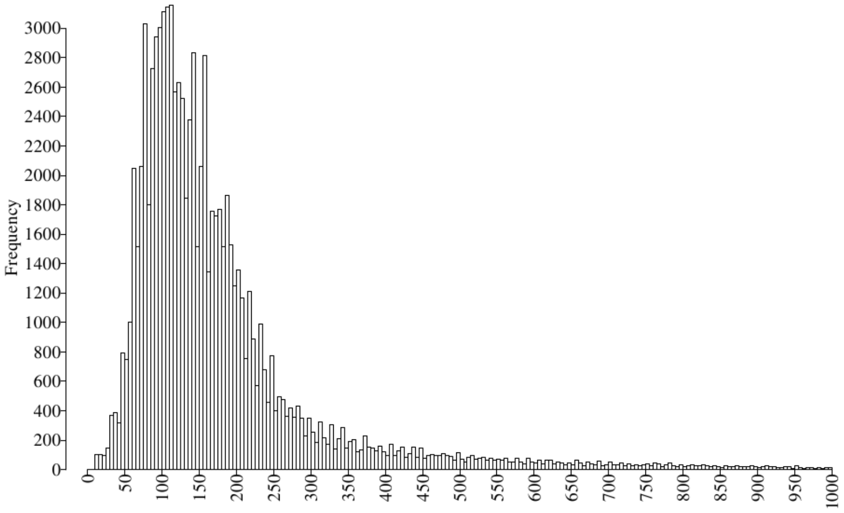


Fig. 3. Histogram of digraph times (up to 1 second)

A banner with the legend -----END---- with the date and time flags is appended when the session ends.

3.2 Statistical Features

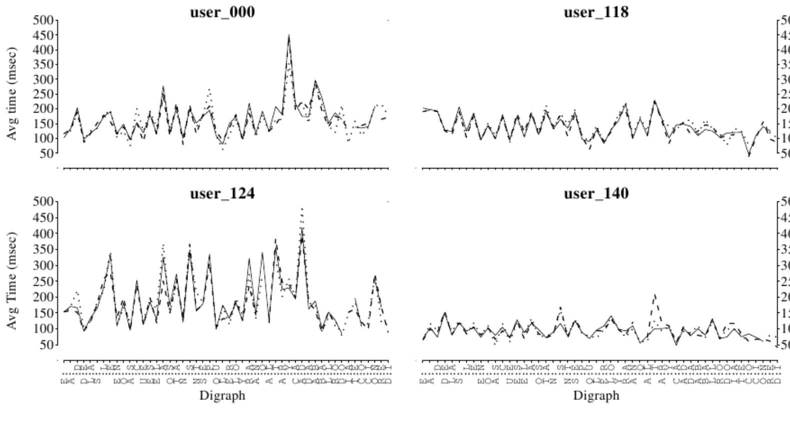


Fig. 4. Example of 4 keystroke paterns grouped in 3 sets of 5 sentences

[6]http://lists.w3.org/Archives/Public/www-archive/2006Nov/att-0047/keyCode-ie.htm.

The dataset contains 282020 keystroke events (key presses and releases); 16372 of them are tagged as invalid. Figure 3 shows that most digraphs were typed in an interval below half a second. Most outliers can be filtered out ignoring those exceeding 500 miliseconds.
Figure 4 shows the average digraph times of 3 fixed groups of sentences for 4 particular users. Only the most frequent digraphs are included. Each user has clearly her own typing pattern, which is consistent throughout different sentences.

4. Future Work

This dataset was collected to perform user identification experiments based on their keystroke pattern. In order to test the effectiveness of the identification/clustering algorithms, a fixed test dataset is the natural first dataset to use. Nevertheless, for identification purposes, free text needs to be used. Therefore, the next step of this work will be the collection of free text keystroke data from labeled users.
On the other hand, this dataset was designed with Spanish texts because most volunteers are Spanish speaking people. A next step could include the collection of the same type of keystroke data in another language.

References

1. Gaines, R., Press, S., Lisowski, W., Shapiro, N. (1980). Authentication by keystroke timing. Rand Report R-256-NSF. Rand Corporation.
2. Monrose, F., Rubin, A. (1997). Authentication via keystroke dynamics. In: Proceedings of the Fourth ACM Conference on Computer and Communications Security, Zurich, Suiza, 48-56.
3. Monrose, F., Rubin, A. (2000). Keystroke dynamics as a biometric for authentication. Future Generation Computer Systems 16(4), 351-359.
4. Monrose, F., Reiter, M., Wetzel, S. (1999). Password hardening based on keystroke dynamics. In: Proceedings of the Sixth ACM Conference on Computer and Communications Security. 73-82.
5. Obaidat, M., Sadoun, B. (1997). Verification of computer users using keystroke dynamics. IEEE Transactions on Systems, Man, and Cybernetics, Part B 27(2), 261-269.
6. Joyce, R., Gupta, G. (1990). Identity authentication based on keystroke latencies. Communications of the ACM 33(2), 168-176.
7. Jiang, C., Shieh, S., Liu, J. (2007). Keystroke statistical learning model for web authentication. In: Proceedings of the 2nd ACM symposium on Information, computer and communications security, ACM New York, NY, USA, 359-361.
8. Rodrigues, R., Yared, G., do N. Costa, C., Yabu-Uti, J., Violaro, F., Ling, L. (2006). Biometric access control through numerical keyboards based

on keystroke dynamics. Lecture notes in computer science 3832, 640.

9. Clarke, N., Furnell, S., Lines, B., Reynolds, P. (2004). Application of keystroke analysis to mobile text messaging. In: Proceedings of the 3rd Security Conference, Las Vegas, NV, 14-15, April.
10. Clarke, N., Furnell, S. (2005). Authentication of users on mobile telephones-a survey of attitudes and practices. Computers & Security 24(7), 519-527.
11. Hwang, S., Cho, S., Park, S. (2009). Keystroke dynamics-based authentication for mobile devices. Computers & Security 28(1-2), 85-93.
12. Zahid, S., Shahzad, M., Khayam, S., Farooq, M. (2009). Keystroke-based user identification on smart phones. In: 12th International Symposium on Recent Advances in Intrusion Detection - RAID, Sep. 2009, Saint-Malo, Francia.
13. Zamonsky, G., Sznur, S. (2009). Keystroke dynamics aplicado a la clasificación de intrusos. In: Workshop de Seguridad Informática - WSegI 2009, Ago. 2009, Mar del Plata, Argentina, SADIO.
14. Maxion, R. (2003). Masquerade detection using enriched command lines. In: International Conference on Dependable Systems and Networks. Volume 0, Los Alamitos, California, USA, IEEE Computer Society, 5.
15. Schonlau, M., DuMouchel, W., Ju, W., M. Theus, A., Vardi, Y. (2001). Computer intrusion: Detecting masquerades. Statistical Science 16, 58-74.
16. Wan, M., Wu, H., Kuo, Y., Marshall, J., Huang, S. (2008). Detecting masqueraders using high frequency commands as signatures. In: Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - AINAW '08; Workshops, Washington, DC, USA, IEEE Computer Society, 596-601.
17. Bertacchini, M., Fierens, P. (2007). Preliminary results on masquerader detection using compression based similarity metrics. Electronic Journal of SADIO 7(1).
18. Bertacchini, M., Benitez, C. (2007). NCD based masquerader detection using enriched command lines. In: Proc. of the IV Congreso Iberoamericano de Seguridad Informática (CIBSI '07), Mar del Plata, Argentina, 329-338.
19. Benitez, C., Fierens, P. (2009). Command dimension reduction in masquerader detection. In: V Conferencia Iberoamericana en Seguridad Informática, CIBSI 2009, Montevideo, Uruguay, Nov. 16-18.
20. Greenberg, S. (1988). Using unix: Collected traces of 168 users. Technical Report 1988-333-45, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada.
21. Schonlau, M. (1998). Masquerading user data. http://www.schonlau.net/intrusion.html.
22. Chinchani, R., Muthukrishnan, A., Chandrasekaran, M., Upadhyaya, S. (2004). RACOON: Rapidly Generating User Command Data for Anomaly Detection from Customizable Templates. In: Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC '04), Tucson, Arizona.

23. Killourhy, K., Maxion, R. (2009). Comparing anomaly-detection algorithms for keystroke dynamics. In: IEEE/IFIP International Conference on Dependable Systems & Networks, 2009, DSN ’09, 125-134.
24. (Master’s thesis)
25. Bergadano, F., Gunetti, D., Picardi, C. (2002). User authentication through keystroke dynamics. ACM Transactions on Information and System Security (TISSEC) 5(4), 367-397.
26. Gunetti, D., Picardi, C. (2005). Keystroke analysis of free text. ACM Transactions on Information and System Security (TISSEC) 8(3), 312-347.
27. Loper, E., Bird, S. (2002). Nltk: the natural language toolkit. In: Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics, Morristown, NJ, USA, Association for Computational Linguistics, 63-70.
28. Greenberg, S. (1988). Using unix: Collected traces of 168 users. Research Report 88/333/45, Department of Computer Science, University of Calgary, Alberta, Canada.
29. Schonlau, M., DuMouchel, W., Ju, W., Karr, A., Theus, M., Vardi, Y. (2001). Computer Intrusion: Detecting Masquerades, Statistical Science (submitted).

Reducing the LSQ and L1 Data Cache Power Consumption

R. APOLLONI[1], P. CARAZO[2], F. CASTRO[3], D. CHAVER[3], L. PINUEL[3] AND F. TIRADO[3]

[1] Universidad Nacional de San Luis, Argentina.
[2] Universidad Politécnica de Madrid, España.
[3] Universidad Complutense de Madrid, España.

Abstract. *In most modern processor designs, the HW dedicated to store data and instructions (memory hierarchy) has become a major consumer of power. In order to reduce this power consumption, we propose in this paper two techniques, one to filter accesses to the LSQ (Load-Store Queue) based on both timing and address information, and the other to filter accesses to the first level data cache based on a forwarding predictor. Our simulation results show that the power consumption decreases in 30-40% in each structure, with a negligible performance penalty of less than 0.1%.*

1. Introduction

Power dissipation in an out of order microprocessor is spread across different structures including Caches, Register Files, the Branch Predictor, the Load-Store Queue, etc. Specifically, the HW dedicated to store data and instructions (the LSQ, the different cache levels, and the main memory) consumes a significant part of the overall power.
In this paper we intend to reduce the LSQ and L1 data cache (DL1) power consumption in an out of order processor. It can be argued that this research problem is not a major concern now due to the trend towards multi-core architectures made by the industry, in which in some cases the pipelines employed are simpler. However homogeneous multi-manycore architectures with in-order pipelines will only provide substantial benefits for scalable applications/workloads, and some researchers have recently highlighted that future designs will benefit from asymmetric architectures that combine simple and power-efficient cores with a few complex and power-hungry cores [1]. The local inefficiencies of a complex core can translate into global performance/per-watt improvements since a complex core could accelerate the serial phases of applications when the power-efficient cores are idle. This way, a single chip will be able to provide good scalability for parallel applications as well as ensure high serial performance. In summary, as promoted in [2], researchers should still investigate methods of improving sequential performance despite we have entered into the multicore era.

We propose two separate techniques for reducing the LSQ and the DL1 power consumption: For the first, based on a set of registers and a small table, we filter many of the required associative accesses that loads and stores would have to perform to the LSQ. For the second, based on a small forwarding predictor, we filter many of the load accesses to DL1. Our techniques lead to high power savings in those structures, which translate into important power savings on the whole processor.
The rest of the paper is organized as follows. Section 2 recaps related work. Section 3 reviews the conventional implementation and brings in our new mechanisms. Section 4 details our experimental environment, while Section 5 outlines experimental results and analyses. Finally, Section 6 concludes.

2. Background

In the last years, there has been a lot of research focused on reducing the LSQ and the Cache power consumption. Next, we summarize the proposals directly related to the mechanisms developed in our job.
Concerning the LSQ power reduction, Sethumadhavan et al. [3] propose an address-based filtering scheme named *search filtering*, which uses hashing to reduce the number of lookups to the LSQ. For this purpose two Bloom Filters [4] are employed, and based only in address information they are able to significantly reduce the associative searches needed while maintaining the program semantics. On the other hand, in our previous work from [5], we introduced two timing-based filtering mechanisms for the LSQ that avoid many of the associative searches that a conventional processor performs unconditionally. The design is based on two sets of age registers: one that filters lookups to the LQ (Load Queue), and another that carries out the same operation over the SQ (Store Queue). Upon execution, just based on straightforward age comparisons between memory instructions and the corresponding register, the mechanism is able to deliver high filtering efficiency with a negligible hardware cost.
Concerning the DL1 power reduction, Nicolaescu *et.al* [6] propose to avoid the data cache access for those loads that receive their data through forwarding. To increase forwarding, they modify the LSQ design to retain load and store instructions after their commit. Thereby, a later load increases its chances of receiving its data from a previous instruction, either an in-flight store, a commited store, or a commited load. The mechanism -named *cached load store queue*, *CLSQ*- is based on the low observed rates of LSQ occupancy for some program phases, that make it possible to earmark unoccupied entries to already commited load or store instructions.

3. Hybrid LSQ Filtering Mechanism

We present here a full LSQ filtering HW, built upon our proposal from [5] and Sethumadhavan's proposal from [3]. In both papers, the main idea is to add simple HW capable of ruling out some memory ordering violations and store to load forwardings. For that purpose, in the first scheme timing information was mainly used, while the second one employed just address information of memory instructions. We combine both in a new hybrid design that provides more filtering capability. Besides, we test our mechanism in a different microarchitectural model –an x86 architecture- than that of the prior works. This model, besides of resulting more appealing, enables for new types of filtering which lead to extra power savings. We should highlight that it includes two new characteristics that are very important for our job[1]: (1) Each store accesses the SQ at issue time in order to perform store-store forwarding; (2) It provides a dependence predictor (LSAP), consisting on an associative table accessed by each load at issue time, that predicts whether a load will alias with a previous store.

3.1 LQ Filtering

Regarding stores searching the LQ looking for premature loads, we can take advantage of a set of registers that basically contain information about loads' age (timing information) and include implicitly some information about loads' address. This approach is referred as Multi-YLA (Multi - Youngest issued Load Age) [5]. The multi-YLA by itself provides really good results in terms of filtering capability, and according to our experiments, combining this scheme with the Bloom Filter from [3] reports no significant improvements.

3.2 SQ and Predictor Filtering

Regarding loads searching the SQ looking for previous dependent stores, and similarly to the Multi-YLA, in [5] we proposed to add a set of registers that contain information about stores' age. This approach is referred as Multi-OFS (Multi - Oldest in Flight Store) [5]. On the other hand, in [3] the authors proposed to add two Bloom Filters (BF) that contain address information (instead of timing information). Due to the very complex updating process of the OFSs set in the multi-OFS scheme its usage is inappropriate. On the other hand, using only a Bloom Filter loses the opportunity of filtering more accesses based on timing information. Hence, we propose to combine a single OFS, that holds timing information and requires a much simpler updating process than a Multi-OFS, and a BF, which provides address-based information. Besides, we include an additional register, called PAS (Pending

[1] More details of the LSQ management in this architecture can be found in [7].

Address Stores), to know if all in flight stores in the processor are resolved. In the next subsections we describe in detail our proposed mechanism.

Filtering of Loads accessing the SQ. When a load instruction issues, it consults the OFS (which holds the age of the oldest in-flight store in the processor) and checks the PAS. If the load is older than OFS (i.e. older than the oldest in-flight store), we can assure, based just on timing information, that a store to load forwarding is not needed for that load, and both the BF access and the SQ associative search can be avoided. Otherwise, the load goes to the second stage of our mechanism, the Bloom Filter (a hashing table with one entry per group of addresses that holds the number of in-flight stores to those addresses). If the corresponding BF entry and the PAS register are both zero[2], or if the BF entry is zero and the LSAP does not predict a dependence with a previous store, we can guarantee, based now on address information, that the load can not receive its data via a forwarding, and again the SQ scan can be avoided. If the BF entry is zero, but PAS is bigger than zero and the LSAP predicts a dependence, an SQ search must be performed to find the closest unresolved store. However, this is a simpler and cheaper access than a common associative one, since we do not need to compare addresses. Finally, if the value recorded in the corresponding BF entry is bigger than zero, the normal SQ associative access is carried out.

Filtering of Loads accessing the Predictor. The new architecture we are using allows for new filtering opportunities. One of them is the chance to filter some accesses of load instructions to the LSAP. The first opportunity to avoid such lookups happens when a load checks the OFS: If the load is older it means that no previous stores exist; hence the LSAP information is irrelevant and the predictor search turns unnecessary.
In order to increase the LSAP filtering capability, we can take advantage of the PAS register: If it is zero, we can also aviod the LSAP access, since all stores are resolved and therefore there is nothing to predict. Otherwise, the LSAP search is required.

Filtering of Stores accessing the SQ. Recall that in the new architecture each store checks the SQ in order to find previous stores to the same address. Once again, we can filter some of these searches. When a store instruction issues, it compares its age with the OFS value. If they are equal, we can guarantee that no prior stores exist, and therefore both the BF access and the SQ lookup can be avoided. Otherwise, the store consults the BF, hashing its address. If the corresponding entry and PAS[3] are zero, we can assure, based

[2] PAS being zero means that we do not need to pay attention to LSAP since no unresolved stores exist.

[3] Note that we use the PAS register in combination with the BF for being able to filter some of the SQ accesses. The BF provides information about resolved stores, while the PAS register reports information about unresolved ones. As we mentioned before, without this register the BF can not be trusted in this context.

now on address information, that this is the only one store to that address, and the SQ search can be avoided. If the BF entry is zero but PAS is bigger than zero, we have to perform an SQ lookup to find the closest unresolved store. Again, this is a simpler and cheaper access than a normal associative search. Finally, if the entry is bigger than zero, a normal SQ lookup is carried out.
This proposed hybrid mechanism exhibits several advantages compared to the multi-OFS [5] and the Bloom Filter [3] schemes. First, thanks to the combination of timing and address information, the number of filterings grows significantly, as we will demonstrate in the evaluation section. Second, unlike the multi-OFS scheme, the updating process for our single-OFS is very simple and incurs no power cost: when a store instruction commits, the OFS is just updated with the age of the store accommodated in the contiguous SQ entry. Third, in our approach the Bloom Filter access is avoided when the 1st stage filters the SQ search based on the OFS (note that the OFS access is much cheaper than the BF one). On the contrary, in the BF scheme, the filter is always accessed at load/store issue.

4. DL1 Filtering using a Forwarding Predictor

4.1 Rationale

In most conventional microprocessors each load instruction consults the first level data cache in order to move the required data into an available register. In parallel, the Store-Queue is searched looking for a previous matching in-flight store. If it is found, the store forwards the corresponding data. Otherwise, the data is provided by the cache. The technique that we propose in this paper is based on the observation that if a load gets its data directly from an earlier store, the data cache access becomes completely unnecessary, and hence we could avoid it saving some power. Obviously, this is only useful if the percentage of loads that get the data from the SQ is high enough.
In a RISC processor, the amount of architectural registers is commonly set to 32 and a register-register architecture is generally implemented. With such configuration, the number of store to load forwardings is relatively small (for example, in [8], less than 15% on average), and maybe the benefits of trying to avoid the DL1 access in such reduced occasions could turn meaningless. However, in a register-memory architecture with only 16 architectural registers -as in the case of x86-64, the architecture employed in this job- the number of store to load forwardings is higher as a result of the extra operations due to register spilling.
In a complementary way, we can use Nicolaescu's CLSQ from [6], which significantly increases the number of loads that receive their data via forwarding, both due to store-load forwarding from the Cached-SQ and to load-load forwarding from the Cached-LQ.
In summary, on a x86-64 architecture using Nicolaescu's Cached-LSQ, the number of forwardings can be relatively high -up to 40% of the loads-, which makes our initial intuition appealling. However, in order to be able to filter

out these accesses, we need to either serialize the LSQ and DL1 cache searches, or know in advance -i.e. make a prediction- whether the load will receive the data via forwarding or not. This is a key issue that has to be addressed.

4.2 Overall structure

As we have just mentioned, an obvious implementation would be to serialize the accesses (as Nicolaescu in [6]): the load first scans the SQ, and then -only when neccessary- the cache is accessed. However, this design is not effcient: when a previous matching store is not found the delay incurred in accessing to the data cache will result in a significant slowdown. In this paper we will turn up with a much more convenient approach.
The design that we propose is based on a forwarding predictor: for each load, we predict whether it will receive its data through forwarding. For convenience of discussion, we loosely refer to these loads as *predicted-dependent* loads and the remainder *predicted-independent* loads. For predicted-dependent loads, only the SQ and the cached-LQ are accessed, omitting the DL1 access (of course, at the risk of being wrong, in which case the cache access is launched with a delay of 1 cycle). For the remaining, both the SQ, the cached-LQ and the DL1 are accessed in parallel (note that in this case, if the predictor is wrong, the data cache access is unnecessary). A predictor with high accuracy provides significant power savings at the cost of a tiny performance degradation. This idea has been explored in similar, yet different contexts [9].
There is a whole lot of research in the field of memory dependence prediction. However, they all employ sophisticated predictor structures, which are excessive for our goal of predicting in advance if a load will receive its data through forwarding. For this reason, we have not considered them in our job. Instead, we have evaluated two kinds of simple predictors: Bloom Filter based [4] and Branch Predictor based [10].

Bloom Filter based predictor. In this first kind of predictors, we implement a low-overhead hash table of counters: At issue time, every load and store hash their memory addresses to a single entry and increment the corresponding counter. Then, at commit, the entry is decremented. Besides, at issue time, loads read the counter before it was incremented to perform the prediction. If it is bigger than zero, there is a likely (but not certain) address match with another memory instruction, and the load is predicted to receive its data via a forwarding. On the other hand, if the counter is zero, the load is predicted-independent.

Branch Predictor based. The second kind of predictors is based on the well-known bimodal branch predictor. Similarly to branch instructions, the majority of loads are usually strongly biased, so such a predictor works well. An advantage of this Bimodal Predictor versus the Bloom Filter based is that the prediction can be performed as soon as the load instruction is decoded, based on its PC. On the contrary, a Bloom Filter is consulted with the memory address of the load, that needs to be calculated first, so the prediction is delayed to issue phase in this case.

Combined Predictor. Finally, we should mention that we have also considered in our evaluation a combined predictor, merging a Bloom Filter with a Bimodal predictor. For extracting the final decision, we predict that a load will receive its data through forwarding only when both structures predict the load to be dependent. Such a structure benefits from both the past forwarding information of loads and memory address information, giving the best results as we will show in the Evaluation Section.

5. Experimental Framework

We have evaluated our proposed design using the PTLsim [11], a performance-oriented simulation tool. The microarchitecture models the default PTLsim configuration that results from the merging of different features of an Intel Pentium 4 [12], an AMD K8 and an Intel Core 2 [13]. Some of the main simulation parameters are listed in Table 1.

Branch predictor	Combined (Bim-2bits + Gshare), 2K BTAC
Instruction Fetch queue size	32
ROB size	128
LSQ size	80 (LQ: 48, SQ: 32)
LSAP size	16
Physical Registers	256
Fuctional Units (INT)	8: 4 ALU (2 INT, 2 FP), 2 Load, 2 Store
Fetch/Decode/Issue/Commit width	4/4/4/4
L1 Instruction Cache L1 Data Cache	32KB (4 way, 64B line) 16KB (4 way, 64B line, 2 cycles latency)
L2 Data Cache L3 Data Cache	256KB (16 way, 64B line, 6 cycles latency) 4MB (32 way, 64B line, 16 cycles latency)
Main memory latency	140 cycles

Table 1. Simulation parameters for default PTLSim configuration

The evaluation of our proposal has been performed using 23 benchmarks from the SPEC CPU2006 suite, compiled for the x86 instruction set. We simulate regions of 100M instructions after reaching a triggering point [14] , that marks the beginning of code area in which the application behavior is representative of the overall execution. To evaluate the impact of our designs over the power consumption of the LSQ or the DL1, we use a power model developed in [14] and CACTI 5.3 [15].

6. Evaluation

6.1 LSQ Power Savings

In this section we report results for the power savings obtained in the LSQ, according with the Power Model explained in [14]. We show 3 different schemes: our proposed scheme (that includes a Multi-YLA for LQ filtering and our Hybrid scheme for SQ and LSAP filtering), the proposal from [3] (that includes two Bloom Filters, one for LQ filtering and the other for SQ filtering), and the proposal from [5] (that includes a Multi-YLA for LQ filtering and a Multi-OFS for SQ filtering). Note that none of these schemes affects performance, since they just care about filtering unnecessary accesses. Figure 1 compares the three approaches for a similar extra HW amount. It shows the average over the studied 23 SPEC-06 applications. Clearly, our scheme reports the highest dynamic power savings. For example, with 1024 bits, the 2 Bloom filters save around 25%, the Multi-YLA + Multi-OFS, 22%, and our design, 38% of the dynamic power consumption of a conventional LSQ. Note that we are including in the LSQ power consumption the LSAP power cost -of course, apart from LQ/SQ power waste-.

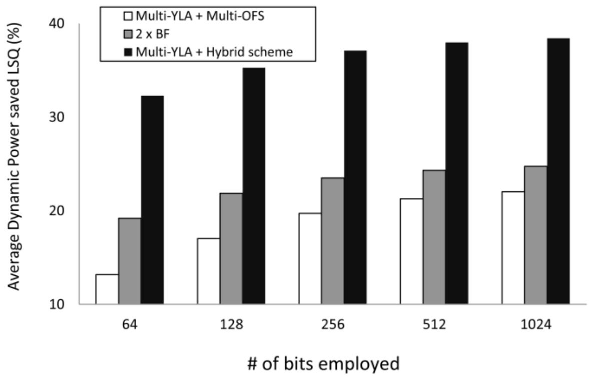


Fig. 1. Average dynamic power saved over the conventional LSQ

6.2 DL1 Power Savings

In this section we show the data cache power reduction and the whole system performance using either the baseline or our alternative. Figure 2(a) shows the power savings achieved in the data cache in our technique with respect to the original architecture. Figure 2(b) illustrates the performance impact of our proposal with respect to the original architecture. In these experiments we always employ the combined predictor, since it reports the highest accuracy values (in [16] we show a comparison of the different forwarding predictors). We can extract the following conclusions.

First, by including our proposed scheme, a significant fraction of loads are correctly predicted-dependent, and therefore the corresponding data cache accesses avoided. This leads to a significant fraction of the DL1 dynamic power consumption eliminated, as Figure 2(a) shows. On average, for a Bloom Filter with 64 entries and a Bimodal Predictor of 256, the DL1 power savings of our approach are around 36%.

Second, and more important, in our architecture average performance remains almost untouched (around 0.1% of slowdown), something that would not happen with Nicolaescu's Proposal. The reason is that in his case, when a load finds no previous dependent stores in the LSQ (i.e. experiments no forwarding) incurs a delay of 1 cycle accessing the DL1, while in our case the forwarding predictor avoids this to happen by predicting most of these loads as independent.

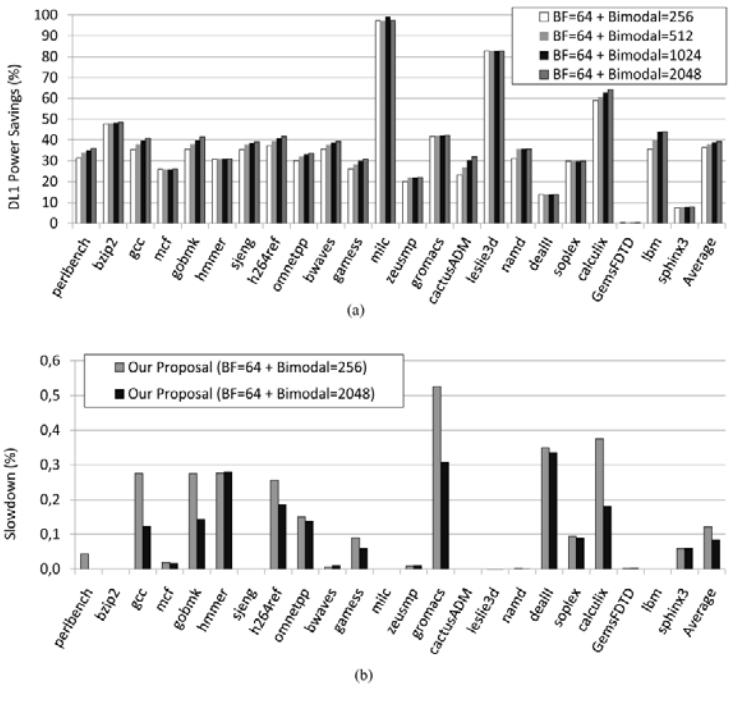


Fig. 2. (a) DL1 Power Savings. (b) Performance Impact

7. Conclusions

The main contributions of this paper are:

- We implement a hybrid design, based on two previous proposals [5] and [3], that filters most of the irrelevant searches to the LSQ. The extra

HW involved is almost negligible and the technique carries no performance degradation at all. On average, our technique saves up to 39% of the LSQ dynamic power consumption.

— We implement a mechanism that filters many accesses to the first level data cache based on a forwarding predictor and Nicolaescu's CLSQ [6]. Both the extra HW and the performance degradation are negligible. On average, our mechanism saves up to 36% of the DL1 dynamic power, with a HW cost less than 100B and a slowdown less than 0.1%.

— All these schemes were tested in a different and more common microarchitectural model -the widespread x86-64- than the one used in previous works.

References

1. Bower, F., Sorin, D., Cox, L. (2008). The impact of dynamically heterogeneous multicore processors on thread scheduling. Micro, IEEE 28(3), May-June, 17-25.
2. Hill, M.D., Marty, M.R. (2008). Amdahl’s law in the multicore era. IEEE Computer 41(7), 33-38.
3. Sethumadhavan, S., Desikan, R., Burger, D., Moore, C., Keckler, S. Scalable Hardware Memory Disambiguation for High ILP Procs. In: MICRO ’03, 399-410.
4. Bloom, B. (1979). Space/Time Trade-offs in Hash Coding with Allowable Errors. Communic. of the ACM 13(7), 422-426.
5. Castro, F., Chaver, D., Pinuel, L., Prieto, M., Tirado, F. (2009). Using Age Registers for a simple Load Store Queue Filtering. Journal of Systems Architecture 55(2), February, 79-89.
6. Nicolaescu, D., Veidenbaum, A., Nicolau, A. Reducing Data Cache Energy Consumption via Cached Load/Store Queue. In: ISLPED ’03, 252-257.
7. Yourst, M. (2007). PTLsim Users Guide and Reference: The Anatomy of an x86-64 Out of Order Superscalar Microprocessor, http://www.ptlsim.org /documentation.php.
8. Castro, F., Chaver, D., Pinuel, L., Prieto, M., Huang, M., Tirado, F. (2006). A Load-Store Queue Design based on Predictive State Filtering. Journal of Low Power Electronics 2(1), April, 27-36.
9. Sha, T., Martin, M., Roth, A. Scalable Store-Load Forwarding via Store Queue Index Prediction. In: MICRO ’05, 159-170.
10. McFarling, S. (1993). Combining Branch Predictors. Technical report tn-36, Western Research Laboratory, Digital Equipment Corporation, June.
11. Yourst, M.T. PTLsim: A Cycle Accurate Full System x86-64 Microarchitectural Simulator. In: ISPASS ’07, 23-34.
12. Hinton, G., Sager, D., Upton, M., Boggs, D., Carmean, D., Kyker, A., Roussel, P. (2001). The Microarchitecture of the Pentium 4 Proc. Intel Technology Journal (Q1 2001).

13. Copenhagen Univ. College of Eng. (2009). The Microarch. of Intel and AMD CPU's: an Optimization Guide for Assembly Programmers and Compiler Makers.
14. Apolloni, R., Chaver, D., Castro, F., Pinuel, L., Prieto, M., Tirado, F. (2010). A hybrid timing-address oriented LSQ filtering for an x86 architecture. Accepted for publication on journal IET-Computers and Digital Techniques.
15. http://www.hpl.hp.com/research/cacti/
16. Carazo, P., Apolloni, R., Castro, F., Chaver, D., Pinuel, L., Tirado, F. (2010). L1 Data Cache Power Reduction using a Forwarding Predictor. Accepted for publication on international conference PATMOS-2010.

II

Innovation in Software Systems Workshop

JAUS Interface for Tools Development in the robotics field

FEDERICO BAZÁN[1], MAURICIO JOST[1], ORLANDO MICOLINI[1]
AND LADISLAO MATHE[2]

[1] Computer Architecture Laboratory, Faculty of Exact, Physical and Natural Sciences,
Universidad Nacional de Córdoba, Argentina.
{bazanfedericoa, mauriciojost}@gmail.com, omicolini@compuar.com.
[2] Group of Robotics and Integrated Systems, Faculty of Exact, Physical and Natural Sciences,
Universidad Nacional de Córdoba, Argentina.
mathe@ieee.org.

Abstract. *Software modularization presents important advantages when it comes to robotic development because of definition of well-known limits for each module that make parallel development easier and reutilization of modules in more systems.*
Components Based Software paradigm follows the mentioned idea and needs definitions that include interfaces, responsibilities and communication rules between components. JAUS standard makes those definitions in the specific context of robotics. There are several SDKs, but they either lack a multi-platform support, or work with platforms whose features do not satisfy the needs of the developers (agile development of proof of concepts, existent projects, community support, among others).
This work extends the possibilities of component implementation, and allows the developer to take advantage of the benefits of the chosen platform. As a result, projects based on widely used tools used in robotics (such as MATLAB, Simulink and LabVIEW) can be easily integrated in a short period of time.

Keywords. *JAUS, OpenJAUS, I-JAUS, MATLAB, Simulink, LabVIEW, manipulator, robotic arm software, SDK, toolbox, component.*

1. Objective

This paper aims to: extend the set of RDTs (Robotics Development Tools) supported for the implementation of JAUS components and facilitate their integration.

2. Theoretical Framework

JAUS (Joint Architecture for Unmanned Systems [1]) is a standard promoted by the United States Department of Defense (OUSD) earmarked to software for unmanned vehicles. JAUS defines component architecture. This means that establishes rules and restrictions for communication between components, features of each, their interfaces, and the group they belong to.

3. Problem statement

In the field of robotic manipulators, software development has a high degree of reusability: recurrent blocks exist such as interface to sensors, closed-loop control blocks, and servos handling blocks, among others. For this reason, component-based software is a convenient alternative.

JAUS materializes the benefits of the components in the field of robotics. There are different JAUS SDK (Software Development Kit) that facilitate the implementation of a component. They are mainly:
OpenJAUS[1], Jaus ++[2], RESquared[3], RI-JAUS[4].

To develop a component using these tools is necessary to program in C / C + +. However, there are higher level languages that provide benefits to the development of robots such as: facilities for the development of GUIs, support for a variety of peripherals such as data acquisition devices and cameras, modeling and simulation toolboxes, etc. These environments are usually the best choice for concept testing, and even final versions of the software of a robot. MATLAB, Simulink, and LabVIEW are the main RDT providing these benefits. In particular there is an alternative designed by Ruel R. Faruque [4], it is limited only to the implementation of JAUS components under LabVIEW.

In this context, the user can choose which platform to develop their components, and then make them interact. However he will only have on his disposal languages like C / C + + and LabVIEW. This represents a serious limitation since there are a large number of developments for MATLAB that could be reused in this area. It is then important the possibility to be able to include MATLAB between the alternatives of JAUS components implementation.

On the other hand, the same working group could develop two different components in different environments using different JAUS SDK, which cannot even rely on the same paradigm. This multiplicity of environments and SDK would represent an extra difficulty for programmers. So it is clear that there is an advantage if all environments use the same SDK as the developer would have to familiarize only with one.
This work is based on those ideas to offer a solution.

[1] http://www.openjaus.com/
[2] http://sourceforge.net/projects/active-ist/
[3] http://www.resquared.com/JAUS-SDK.html
[4] http://www.repinvariant.com/dist/ri-jaus/0.9.0beta/ri-jaus.html

4. Raising Solution

This propounds the implementation of an cross-platform interface that allows the development of JAUS components from any of the above RDT. This scenario allows to continue to develop common solutions in the RDT, and integrate them through the standard without much effort.

The solution has a dynamic link library (DLL) developed in C language, which has been called JAUS Interface (I-JAUS[5]). This interface makes use of a SDK called OpenJAUS. Its justification can be seen in [6].

I-JAUS offers the developer a set of calls, as they are the main OpenJAUS defined which can be called from MATLAB or LabVIEW. They are:

- Initialize an OpenJAUS component.
- Add services to the component.
- Start the component.
- Update its outgoing messages.
- Send these outgoing messages.
- Get incoming messages.
- Finish the OpenJAUS component.

Fig. 1 shows a component developed with I-JAUS as part of aJAUS system.

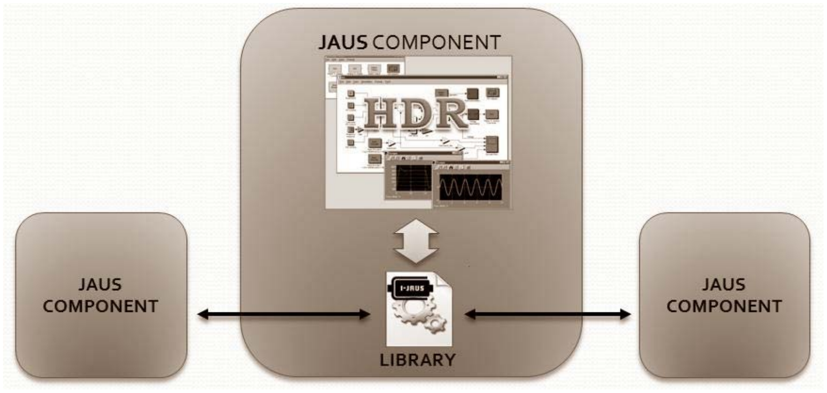


Fig. 1. JAUS Component implemented in RDT

[5] http://www.code.google.com/p/i-jaus/

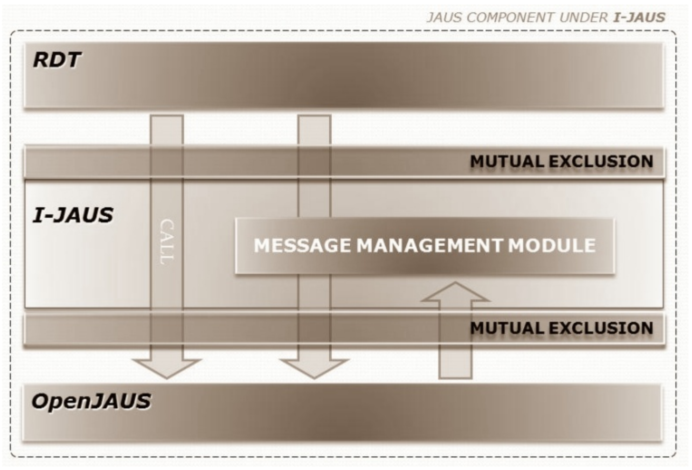


Fig. 2. I-JAUS architecture

Fig. 2. Shows the layered architecture of a component developed through IJAUS. This interface acts as mediator between the demands of the RDT and the OpenJAUS component.

4.1. Concurrency problema

OpenJAUS allows to define an user structure for each component. It is used for messages storage and all the information deemed necessary. In addition, a component features states and functions associated with each. When implementing an OpenJAUS component, one thread is started in charge of the processing associated with each state and the message processing. Running a component from the thread of the RDT a new OpenJAUS thread is started. This new thread aims to deal with the asynchronous arrival of messages. This is done to:

- Decoupling the processing of incoming messages made by OpenJAUS (asynchronous reception) and the user code processing on the RDT. This allows lower latency when generating calls from the RDT itself.
- The library handles internally the blocking calls, and with this new thread it avoids blocking the RDT. Reciprocally, the RDT could be blocked without blocking the OpenJAUS thread (and do not receive messages for it).

With the execution of two threads it is possible to simultaneously read and modify any data or message in the user structure of a component. This creates situations prone to loss of integrity. This problem is solved by a mutual exclusion mechanism, which is applied to each of the calls involved in message processing.

4.2 Treatment of incoming messages

The threads of RDT and OpenJAUS have a data stream associated with the messages you want to send or receive. This flow of data implies a link between two asynchronous entities, which is resolved through a message queue.
In cases in which the processing capacity of incoming messages is not enough, there would be an overload in the queue (potential loss). Since JAUS mostly defines components to conform a control system, discard policy must ensure the preservation of updated data. However, it is important to note that JAUS defines a Sequence Number field and an ACK field in the header of each message. Both can be used if a traffic control is necessary.
Figure 2 shows the Message Management Module, which implements the mentioned policy. This module was designed so that a component keeps the last copy of each type[6] of incoming message. It also supports a dynamic priority mechanism based on the order of arrival of message types. This prevents the occurrence of starvation.

5. Testing

As part of the I-JAUS proyect, a set of experiments have been developed that exemplify the use of the interface. These show that it is possible to adapt to the standard projects implemented in multiple languages.
To control the manipulators through JAUS commands, Subsystem Component Commander (SSC) were created that uses projects that can interact with the user through: mouse, keyboard, GUI, joystick, voice or eye tracking. These SSC assume the role of Operator Control Unit (OCU) and give commands to the component that envelops the manipulator.

5.1 Example of integration of components based on multiple RDT

The PUMA3D Project (MATLAB) has been involved in an End Effector Pose Driver component (EEPD). Because of this it can be controlled by an OCU implemented in LabVIEW. This is shown in Fig. 3.

[6] The message type is determined by ID. For example, two instances of the message Set Joint Positions correspond to the same type.

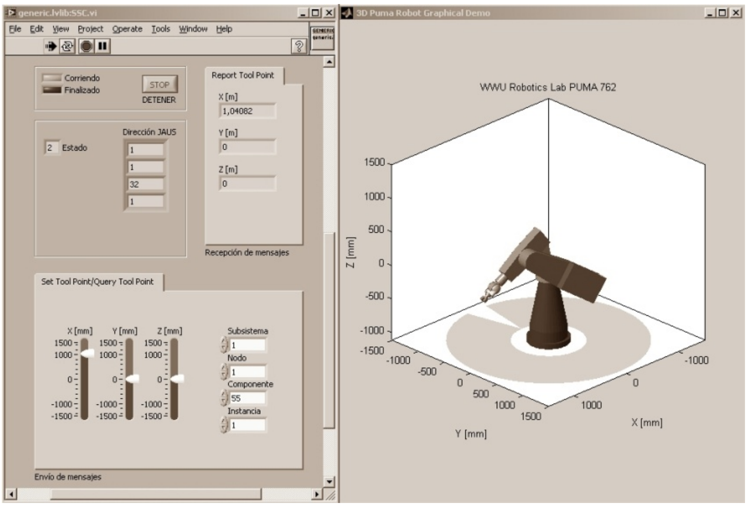


Fig. 3. Components of LabVIEW (SSC Panel) and MATLAB (PUMA3D in EEPD) interacting

The complete set of demonstrations can be seen on the I-JAUS page (http://i-jaus.googlecode.com).

6. Conclusions

Development Environments

Tests have shown that the integration of components implemented in different environments, allows greater flexibility in deployment times. The developer can select the tool according to their features, support, and existing projects that it offers and according to his own skills as well.
In particular, the incorporation of MATLAB provides an user-friendly environment and a variety of mathematical functions associated with the control. However, it is important to emphasize your command interpreter, which has allowed to reduce the time to implement each test performed during IJAUS testing, compared to those made in C. Such interpreter allows to write code in a more dynamic, and then reuse the commands executed through scripts with minimal effort. We can say then that, besides being the most common environment in the studied environment, it provides facilities to further justify its choice.

Proof of concept

Thanks to I-JAUS you can take benefit of a lot of work done in MATLAB and LabVIEW, and integrate them to the JAUS standard in a very simple way and in a short time. For example, all the demonstrations developed for this study were performed in less than two weeks. This agility is the result of two points:

- The easiness of I-JAUS to be integrated into an existing project of an operating RDT.
- The means provided by I-JAUS, the JAUS standard, for projects to communicate with each other.

Impact

I-JAUS has immediate benefits for users. Of those observed, the most important are:

- Dividing a project into blocks. This mobilizes the group to end rigorously tested components. So it is possible to abstract from the implementation of parts that are considered reliable, to continue experimenting on others.
- JAUS learning facilitation. This occur thanks that MATLAB allows interactive use of a JAUS component (developed in I-JAUS) with its command interpreter.
- Increase in considered projects. LabVIEW and MATLAB are cradles of innovative developments now integrated through a component-based standard.

Contributions

This paper provides the user with a dynamic link library (DLL) with the documentation necessary to use it from both MATLAB and LabVIEW. All material is freely distributed under BSD[7] license.

Future

Although I-JAUS is currently based on version 3.3 of the Reference Architecture documents, its update is possible in order to link to the standard improvements.
Since the project has met the stated objective, the fact of extending the functionalities of I-JAUS takes value, and so supports the other component groups of the standard.
In addition of the interface, this work leaves a set of implemented components. This opens the door to each user to improve the collection, create new components and share them with the community.

[7] I-JAUS: http://code.google.com/p/i-jaus.

References

1 Rowe, S. Wagner, C. (n.d.) (2010). An Introduction to the Joint Architecture for Unmanned Systems. USA: Cybernet Systems Corporation. Feb., 1. http://www.openskies.net/papers/07F-SIW-089 IntroductiontoJAUS.pdf.

2. The Joint Architecture for Unmanned Systems. Reference Architecture Specification. Volume II, Parts 1, 2 and 3. Version 3.3. (2007). USA: Jaus Working Group.

3. OpenJAUS. May, 30, 2009. http://openjaus.com/trac/openjaus.

4. Faruque, R. (2010). A JAUS Toolkit for LabVIEW, and a Series of Implementation Case Studies with Recommendations to the SAE AS-4 Standards Committee. EEUU: Virginia Polytechnic Institute and State University. Feb., 1. http://scholar.lib.vt.edu/theses/available/etd-01142007 - 212355/unrestricted/jaus.pdf.

5. Sosa, O. (2004). Design and Implementation of a Modular Manipulator Architecture. USA: University of Florida.

6. Bazán, F. Jost, M. (2010). Interfaz JAUS para Herramientas de Desarrollo de Software de Robots. Argentina: Universidad Nacional de Córdoba. March, 6. http://i-jaus.googlecode.com/files/ijaus_report.pdf.

II

Signal Processing and Real-Time System Workshop

Diffuse Outlier Time Series Detection Technique for Functional Magnetic Resonance Imaging

JAVIER GIACOMANTONE[1] AND TATIANA TARUTINA[2,3]

[1] Instituto de Investigación en Informática (III-LIDI),
Facultad de Informática, Universidad Nacional de La Plata.
jog@lidi.info.unlp.edu.ar.
[2] Instituto de Física, Facultad de Ciencias Exactas,
Universidad Nacional de La Plata.
tarutina@fisica.unlp.edu.ar.
[3] Consejo Nacional de Investigaciones Científicas y Tecnológicas.

Abstract. *We propose a new support vector machine (SVM) based method that improves the time series classification in magnetic resonance imaging (fMRI). We exploit the robust anisotropic diffusion (RAD) technique to increase the classification performance of the one class support vector machine by taking into account the hypothesis of spatial relationship between active voxels. The proposed method was called Diffuse One Class Support Vector Machine (DOCSVM). DOCSVM method treats activated voxels as outliers and applies one class support vector machine to generate an activation map and RAD to include the neighborhood hypothesis, improving the classification and reducing the iteration steps with respect to RADSPM. We give a brief review of the main methods, present receiver operating characteristic (ROC) results and conclude suggesting further research alternatives.*

Keywords. *Time Series, Functional Magnetic Resonance Imaging, classification, Support Vector Machines, Robust Anisotropic Diffusion.*

1. Introduction

The purpose of fMRI is to map areas of increased neuronal activity of the human brain associated with cognitive or motor tasks. The hemoglobin in the blood is a natural contrast agent, because it has different magnetic properties depending of its state of oxygenation. These differences affect the voxel intensity in the magnetic resonance images [1]. Baseline images are scanned periodically while the subject is at rest (or in other baseline condition) and activation images are acquired when the subject is performing a specific task or receiving a stimulus. A fMRI image can be seen as a set of time series where each time series corresponds to one voxel in the structural image. Classification of time series is the main subject of brain fMRI data analysis. A number of different techniques have been developed for fMRI data analysis, and can be classified in two main categories, model driven [2][3][4] and data driven methods [5][6]. Data driven methods use a method in

machine learning or statistics to analyze fMRI time series while model driven methods assume a model related to the structure and function of the brain. Support vector machine (SVM) being a data driven method has been applied to the supervised classification of cognitive states [7] by optimizing a margin and using the kernel trick [8]. One class SVM (OCSVM) [9][10] has been applied to fMRI unsupervised classification [11][12]. Brain fMRI time series on most voxels are independent of the experimental stimulus and time series on only few voxels are related to the experimental stimulus. Time series related to the stimulus can be considered as outliers and time series not related to the experimental stimulus as normal data points, satisfying the hypothesis necessary to apply the OCSVM methods. In order to include the spatial relationship between activated voxels, that assumes that time series on close voxels have similar state activation correlative or irrelative to the experimental stimulus, new alternatives have been proposed [13][14][15][16]. In this work we present preliminary results of a new technique, DOCSVM, that combines the one class support vector machine and the RAD to improve the classification of fMRI time series by considering the neighborhood spatial relationship.
The paper is organized as follows, section 2 to 4 cover the fundamental ideas behind OCSVM, RAD and DOCSVM. Section 5 and 6 show experimental results, conclusions and propose future research work.

2. One-class SVM

There are two main one class classification algorithms based on SVM, support vector data description [9] and one-class SVM [10]. A typical example of interest of one class classification is the outlier detection that attempts to detect uncharacteristic objects from a data set. The one-class SVM estimates a function f that is positive for a subset of the sample space and negative for the complement. The algorithm maps the data into a feature space corresponding to the kernel and separates them from the origin with maximum margin. Different types of kernels can be used corresponding to nonlinear estimators in the input space.
Consider a given data set

$$x_1, \ldots, x_l \in \chi,$$

where $l \in \chi$ is the number of observations and χ is a compact subset of $\square^{N}$. Let $\Phi : \chi \rightarrow \mathrm{F}$ be a feature map, that is, a map into an inner product space F such that the inner product in the image of Φ can be computed by evaluating a simple kernel.

$$k(x,z) = (\Phi(x) \cdot \Phi(z))$$

It can be formulated as an optimization problem.

$$\min_{w\in F,\xi\in\mathbb{R}^l,\rho\in\mathbb{R}} \quad \frac{1}{2}\|w\|^2+\frac{1}{\nu l}\sum_i \xi_i-\rho$$

$$\text{s.}t. \qquad (w\cdot\Phi(x_i))\geq\rho-\xi_i,\xi_i\geq 0, \tag{1}$$

where $\nu\in(0,1]$ is a parameter controlling the penalized term and ξ_i are slack variables. By solving the optimization problem (1) we obtain w and ρ and the decision function is -1 for outliers in the data set and +1 for the rest of the samples in the data set.

$$f(x)=\text{sgn}(w\cdot\Phi(x))-\rho) \tag{2}$$

Introducing Lagrangian multipliers $\alpha_i,\beta_i\geq 0$, we obtain

$$L(w,\xi,\rho,\alpha,\beta) \quad = \quad \frac{1}{2}\|w\|^2+\frac{1}{\nu l}\sum_i\xi_i-\sum_i\beta_i\xi_i-\rho$$

$$- \qquad \sum_i\alpha_i(w\cdot\Phi(x)-\rho+\xi_i)$$

Setting the derivatives with respect to the primal variables w,ξ,ρ equal to zero yields

$$w=\sum_i\alpha_i\Phi(x_i),$$

$$\alpha_i=\frac{1}{\nu l}-\beta_i\leq\frac{1}{\nu l},$$

$$\sum_i\alpha_i=1.$$

The decision function can be written as

$$f(x)=\text{sgn}(\sum_i\alpha_i k(x_i,x)-\rho)$$

The multipliers α_i can be solved from the dual problem:

$$\min_{\alpha} \quad \frac{1}{2}\sum_{ij}\alpha_i\alpha_j k(x_i, x_j)$$

$$\text{s.}t. \quad 0 \ge \alpha_i \ge \frac{1}{\nu l}, \sum_i \alpha_i = 1$$

The parameter ρ can be recovered by exploiting that for any such α_i and the corresponding pattern x_i satisfies

$$\rho = (w \cdot \Phi(x_i)) = \sum_j \alpha_i k(x_i, x_j). \qquad (3)$$

3. Robust Anisotropic Diffusion

Perona and Malik [17] defined the anisotropic diffusion as

$$\frac{\partial I(x,y,t)}{\partial t} = \mathrm{d}iv\left[g\left(\left\|\nabla I(x,y,t)\right\|\right)\nabla I(x,y,t)\right], \qquad (4)$$

using the original image $I(x,y,0) : \mathbb{R}^2 \to \mathbb{R}^+$ as the initial condition, where t is an artificial time parameter and g is an ``edge-stopping" function. The right choice of g can greatly affect the extent to which discontinuities are preserved. Perona and Malik suggested two possible edge-stopping functions in their paper [17]. Black et al. [18] used the robust estimation theory to choose a better edge-stopping function, called Tukey's biweight:

$$g(x) = \begin{cases} \left[1 - \dfrac{x^2}{5\sigma^2}\right]^2, & \dfrac{x^2}{5} \le \sigma^2 \\ 0, & \text{o}therwise \end{cases} \qquad (5)$$

The function g above is the dilated and scaled version of the original Tukey's function, where $g(0) = 1$ and the local maxima of its ``influence function" $\psi(x) = xg(x)$ is situated at $x = \sigma$. The diffusion that uses the Tukey's function is called robust anisotropic diffusion (RAD) and this is the edge-stopping function adopted in this paper.
Perona and Malik [17] discretized spatio-temporally their anisotropic diffusion equation (4) as:

$$I(s,t+1) = I(s,t) + \frac{\lambda}{|\eta_s|} \sum_{p \in \eta_s} g(|\nabla I_{s,p}(t)|)\nabla I_{s,p}(t), \qquad (6)$$

where $I(s,t)$ is a discretely sampled image, s denotes the pixel position in a discrete 2-D or 3-D grid, $t \geq 0$ now denotes discrete time steps, the constant λ determines the rate of diffusion (usually, $\lambda = 1$), and η_s represents the set of spatial neighbors of pixel s. For 2-D images, usually four neighbors are considered: *north*, *south*, *west* and *east*, except at the image boundaries. For 3-D images, six voxels are usually considered: the above-mentioned four plus ``up'' and ``down'' voxels. The gradient magnitude of a voxel in a particular direction at iteration t is approximated by:

$$\nabla I_{s,p}(t) = I(p,t) - I(s,t), p \in \eta_s. \qquad (7)$$

Black et al. [18] suggested to use the ``robust scale'' defined by:

$$\sigma_e = 1.4826\mathrm{M}AD(\vec{\nabla} I) = 1.4826\mathrm{m}edian_I \left[\left| \left\| \vec{\nabla} I \right\| \vec{\nabla} I - \mathrm{m}edian_I (\left\| \vec{\nabla} I \right\|) \right| \right], \ (8)$$

where MAD is the Median Absolute Deviation.

4. Diffuse One-class SVM

By combining OCSVM and RAD we proposed a new technique that improves the classification of fMRI temporal series under the validity of the spatial neighborhood hypothesis.

Let $I^{'}$ be an fMRI data. First of all, the mean value is removed from $I^{'}$, yielding the mean-removed fMRI I :

$$I = I^{'} - \bar{I}^{'} \qquad (9)$$

This pre-processing is very important, because structural and functional regions of the brain do not necessarily match. No structural information should be diffused, but only the activation information. Note that the activation information is not affected at all by the mean-correction.

Time series on each voxel is taken as a data point. The $x_{i,j,k}$ is the data point corresponding to the i^{th} row, j^{th} file and k^{th} slice identifying one particular time series or data point. The $x_{i,j,k}$ are directly the input into the optimization problem (1). The optimal solutions w and ρ can be obtained by solving the dual problem and (3). Then for each $x_{i,j,k}$ a primal decision value $y_{i,j,k}$ is obtained,

$$y_{i,j,.k} = \left(w.\Phi\left(x_{i,j,k}\right)\right) - \rho \qquad (10)$$

that represents the distance between a point $\Phi\left(x_{i,j,k}\right)$ and a hyperplane in the high-dimensional kernel space $\left(w.\Phi\left(x_{i,j,k}\right)\right) - \rho = 0$.

Let us denote the fMRI data at iteration $t \geq 0$ of the diffusion process as $I\left(s,n,t\right)$, where $I\left(s,n,0\right)$ is the initial mean-corrected fMRI at spatial voxel position s and volume n, and $T\left(s,t\right)$ is the activation map form by $y_{i,j,k}\ \forall i,j,k$ where s is a particular position i,j,k.

1. Let $t \leftarrow 0$.
2. Calculate the activation map $T\left(s,0\right)$ by using OCSVM (10).

3. Compute the diffusion coefficients. The diffusion coefficient between a voxel s and its neighboring voxel p at instant t is:

$$g\left(\left|\vec{\nabla} T_{s,p}(t)\right|\right), \mathit{where}\ \vec{\nabla} T_{s,p}(t) = T\left(p,t\right) - T\left(s,t\right) \qquad (11)$$

4. Use these coefficients to perform the diffusion in $I\left(s,n,t\right)$, yielding the diffused fMRI, $I\left(s,n,t+1\right)$, at iteration $t+1$:

$$I\left(s,n,t+1\right) \leftarrow I\left(s,n,t\right) + \frac{\lambda}{\left|\eta_s\right|} \sum_{p \in \eta_s} g\left(\left|\vec{\nabla} T_{s,p}(t)\right|\right) \vec{\nabla} I_{s,p}(t) \qquad (12)$$

where $\vec{\nabla} I_{s,p}(n,t) = I\left(p,n,t\right) - I\left(s,n,t\right)$.

5. Let $t \leftarrow t+1$ and repeat steps 2 to 5 some predefined number of times or until the average of diffused values (second term of equation (12)) is below some predefined threshold.

6. Classify each voxel applying the decision function of equation (2).

The anisotropic diffusion is controlled by the number of iterations and the scale parameter of the edge stopping function (5), σ.

5. Experimental Results

In order to test and develop classification models in fMRI three main data sets are commonly used. The first model is a completely synthetic one, carefully designed to reproduce real fMRI conditions like signal to noise ratio (SNR), type of noise and spatial distribution of activated voxels. The second alternative is to generate artificial foci of activated voxels in real fMRI data. The third possible data set involves working with real fMRI time series. We present comparative results on synthetic data sets known also as synthetic time series, artificial images or phantoms. These types of experiments provide controlled conditions and knowledge of the exact activated region location, namely a gold standard. In order to test and compare results of the proposed method we generate two artificial images, based on the phantom proposed in [19], with different activation levels. The 4D fMRI model is formed by $10 \times 10 \times 3$ voxels per volume and 84 volumes. Voxels values were 16000 corrupted by zero-mean Gaussian noise with standard deviation $\sigma = 4000$. Active voxels had their values increased by 1000 for phantom I and 1500 for phantom II. The fMRI experiment had alternating blocks of 6 non-active and 6 active volumes, beginning with non-active volumes. Activated volumes had a $6 \times 6 \times 3$ activated region in the center of the volume, with two non-activated regions of $2 \times 2 \times 3$ voxels each. Fig. 1 depicts one activated slice of phantom II, the gold standard and the DOCSVM activation map.

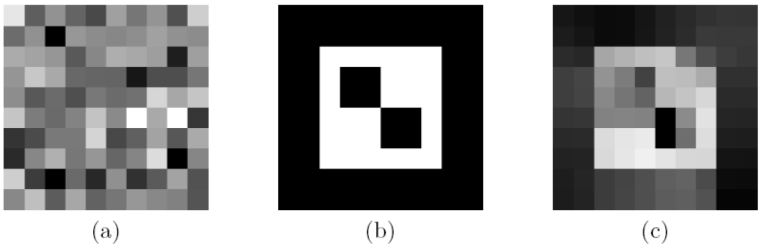


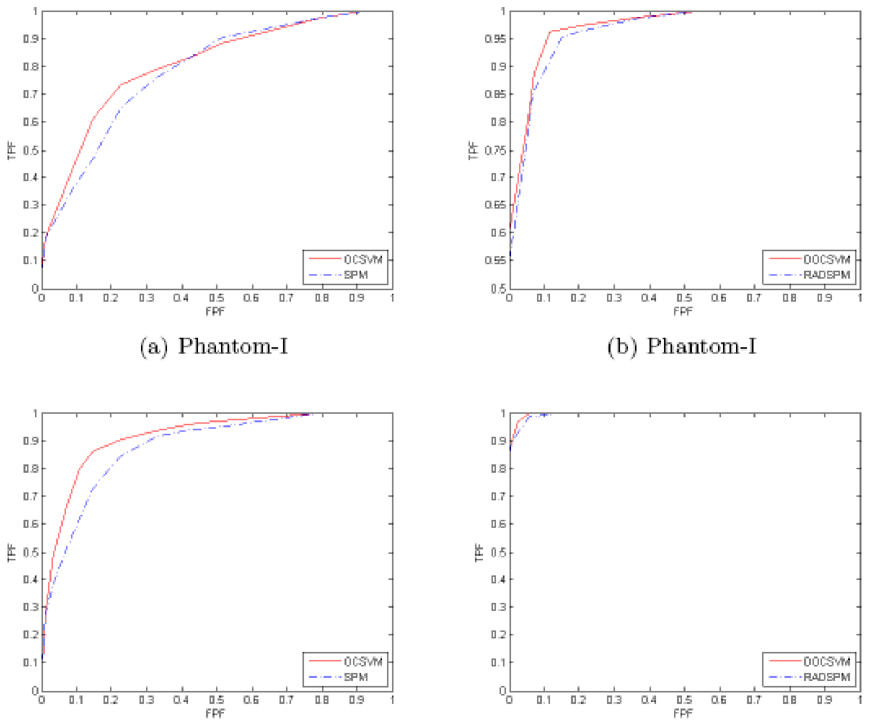
Fig. 1. (a) Simulated fMRI slice, (b) Reference image (gold standard), (c) Activation map produced by DOCSVM ($\nu = 0.7, \sigma = 2, t = 6$)

In the experiments on the two synthetic datasets, the principal parameters of the algorithms were set as follows. The radial basis function was chosen as the kernel function for the OCSVM and $\nu = 0.7$. RADSPM and DOCSVM are sensible to the scale parameter selection [20], beginning with $\sigma = \sigma_s$ we adjust σ using ROC curves as a gauging procedure as suggested in [21].

We have obtained comparative results of four different methods by using the well-known Receiver Operating Characteristics (ROC) analysis [21][22][23]. Let TP, FN, FP and TN be respectively the number of true positives, false negatives, false positives and true negatives obtained by comparing the ideal classification (gold standard) and the results obtained by each of the evaluated methods. Then, the True Positive Fraction (TPF) and the False Positive Fraction (FPF) are defined as:

$$\mathrm{T}PF = \frac{\mathrm{T}P}{\mathrm{T}P + FN}, \quad \mathrm{F}PF = \frac{\mathrm{F}P}{\mathrm{F}P + TN} \qquad (13)$$

Figure 2 depicts correlation's SPM, OCSVM's, RADSPM's and DOCSVM's ROC curves. Each point of a ROC curve is obtained by solving equation (13) for a specific threshold value. Table 1 presents some performance metrics of the four ROC curves, all of them demonstrating the improved performance of DOCSVM and RADSPM compared to the non-spatial oriented methods considered in the experiments. The area under the curve and the distance d_{oop} from the principal diagonal to the optimal operating point (OOP)(the point of the curve most distant from the principal diagonal), are superior for DOCSVM with less diffusion iteration steps with respect to the results of RADSPM.



(a) Phantom-I (b) Phantom-I

(c) Phantom-II (d) Phantom-II

Fig. 2. ROC curves

	Method	Area	d_{po}	TPF_{po}	FPF_{po}
Phantom-I	Correlation-SPM	0.7863	0.3063	0.7619	0.3287
	$RADSPM_{\sigma=1.8,t=10}$	0.9645	0.5687	0.9524	0.1481
	$OCSVM_{\nu=0.7}$	0.8081	0.3591	0.7348	0.2269
	$DOCSVM_{\sigma=1.8,t=8,\nu=0.7}$	0.9716	0.5993	0.9624	0.1148
Phantom-II	Correlation-SPM	0.8798	0.4373	0.8452	0.2269
	$RADSPM_{\sigma=2,t=10}$	0.9958	0.6594	0.9881	0.0556
	$OCSVM_{\nu=0.7}$	0.9166	0.5080	0.8619	0.1415
	$DOCSVM_{\sigma=2,t=6,\nu=0.7}$	0.9975	0.6685	0.9686	0.0231

Table 1. Performance metrics

6. Conclusions and Future Work

In this paper we have presented a new SVM based technique named DOCSVM taking into account the spatial relationship activation hypothesis. This technique improves fMRI time series classification. We compared this method to OCSVM, correlation analysis and RADSPM. Experimental results using ROC curves on synthetic data sets have shown promising results for DOCSVM. The proposed method treats activated voxels as outliers and applies OCSVM to generate an activation map and RAD to include the neighborhood hypothesis, improving the classification and reducing the iteration steps with respect to RADSPM. The obtained results of DOCSVM are preliminary. Further research involves improving the probabilistic model used to create the artificial images considering the noise distribution and the signal to noise ratio in order to approximate the complex and noisy fMRI signal structure. Extensive tests on real fMRI and artificial data must be done in order to adjust parameters and extend our results to different real experiment paradigms.

References

1. Ogawa, S. et. al. (1993). Functional brain mapping by blood oxygenation level-dependent contrast magnetic resonance imaging. Biophysics Journal, 14(3):803-812.
2. Friston, K. J., A. P. Holmes, K. J. Worsley, Poline J. P., C. D. Frith, R. S. Frackowiak (1995). Statistical parametric maps in functional imaging: a general linear approach. Human Brain Mapping, 2:189-210.

3. Faisan, S., L. Throava, J. Armspach, M. Metz-Lutz, F. Heith (2005). Unsupervised learning and mapping of active brain functional MRI signals based on hidden semi-Markov event sequence models. IEEE. Transactions on Medical Imaging, 24(2):263-276.
4. Tian, J., L. Yang, J. Hu. Recent advances in the data analysis method of functional magnetic resonance imaging and its applications in neuroimaging. Progress in Natural Science, 16(8):785-795.
5. Goutte, C., P. Toft, E. Rostrup, F. Nielsen, L. Hansen (1999). On clustering fMRI time series. NeuroImage, 9:298-310.
6. Friman, O., J. Carlsson, P. Lundberg, M. Borga, H. Knutsson (2001). Detection of neural activity in functional MRI using canonical correlation analysis. Magnetic Resonance in Medicine, 45(2):323-330.
7. Cox, D., R. L. Savoy (2003). Functional magnetic resonance imaging (fMRI) brain reading: detecting and classifying distributed patterns of fMRI activity in human visual cortex. NeuroImage, 19:261-270.
8. Vapnik, V. (1999). An overview of statistical learning theory. IEEE Transactions on Neural Networks, 10(5):988-999.
9. Tax, D. M. J., R. P. W. Duin (2004). Support vector data description. Machine Learning, 54:45-66.
10. Schölkopf, B., J. C. Platt, J. Shawe-Taylor, A. J. Samola, R. C. Williamson (2001). Estimating the support of a high dimensional distribution. Neural Computation, 13(7):1443-1471.
11. Wang, D. F., D. S. Yeung, E. C. Tsang (2007). Ellipsoidal support vector clustering for functional MRI analysis. Pattern Recognition, 40(10):2685-2695.
12. Song, X., A. M. Wyrwicz (2009). Unsupervised spatiotemporal fMRI data analysis using support vector machines. NeuroImage, 47:204-212.
13. Chen H. F., D. Z. Yao, S. Becker, Y. Zhou, M. Zeng, L. Chen (2002). A new method for fMRI data processing: Neighborhood independent component correlation algorithm and its preliminary application. Science in China Series, 45(5):373-382.
14. Sole, A. F., S. C. Ngan, G. Shapiro, X. P. Hu, A. López (2001). Anisotropic 2D and 3D averaging of fMRI signals. IEEE Transactions on Medical Imaging, 20(2):86-93.
15. Kim, H. Y., J. Giacomantone, Z. H. Cho (2005). Robust Anisotropic Diffusion to Produce Enhanced Statistical Parametric Map. Computer Vision and Image Understanding, 99:435-452.
16. Yang, J., N. Zhong, P. Liang, J. Wang, Y. Yao, S. Lu (2010). Brain activation detection by neighborhood one-class SVM. Cognitive Systems Reasearch, 11:16-24.
17. Perona, P., J. Malik (1990). Scale space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(7):629-639.
18. Black, M. J., G. Shapiro, D. H. Marimont, D. Hegger (1998). Robust anisotropic diffusion. IEEE Transactions on Image Processing, 7(3):421-432.

19. Kim, H. Y., J. Giacomantone (2005). A New Technique to Obtain Clear Statistical Parametric Map By Applying Anisotropic Diffusion to fMRI. IEEE International Conference on Image Processing, 724-727.
20. Voci, F., S. Eiho, N. Sugimoto, H. Sekiguchi (2004). Estimating the gradient threshold in the perona-malik equation. IEEE Signal Processing Magazine, 39-46.
21. Giacomantone, J., A. De Giusti (2008). ROC performance evaluation of RADSPM technique. Argentinian Congress on Computer Science.
22. Sorenson, J. A., X. Wang (1996). ROC Method for Evaluation of fMRI techniques. Magnetic Resonance in Medicine, 36:737-744.
23. Skudlarski, P., T. Constable, J. C. Gore (1999). ROC Analysis of Statistical Methods Used in Functional MRI: Individual Subjects. Neuroimage, 9:311-329.

Processing Ambiguous Fault Signals with Three Models of Feedforward Neural Networks

SERGIO L. MARTÍNEZ[1], ENRIQUE E. TARIFA[1,2] AND SAMUEL FRANCO DOMINGUEZ[1]

[1] Facultad de Ingeniería, Universidad Nacional de Jujuy,
Gorriti 237, S. S. de Jujuy, Jujuy, Argentina
{smartinez, eetarifa, sfdominguez}@fi.unju.edu.ar
[2] CONICET, Argentina.
eetarifa@arnet.com.ar

Abstract. *In the industrial technological field, running equipment or processes usually is monitored through automatic diagnosis systems. Within several Technologies for implementing such systems, the artificial neuronal networks are the most successful and widely spread. The data signals coming from the equipments or processes under supervision are interpreted by the neuronal networks so as to diagnose the presence of any fault. In this work three models of artificial neural networks and two methods of training are analyzed so as to establish, based on real experiences, the best combination of the neuronal model and the training method for recognizing in an efficient way the ambiguous patterns of faults.*

Keywords: *Neural Networks. Diagnosis. Ambiguous Fault Signals. Optimized training.*

1. Introduction

In the automatic diagnosis of faults in equipments or processes, especially industrial, several sensors associated to these, give data sequences which, precisely interpreted, can reveal the working status, normal or abnormal, of such equipments or processes. Such data are analyzed by a diagnosis system in order to determine the cause (fault) o fan eventual abnormality, so as to establish the necessary correcting actions.
Formally the mission of a diagnosis system is to analyze the status of the process under supervision, so as to determine if any fault (detection stage) has appeared. In such case, the diagnosis system must analyze how is the evolution of the process status so as to identify the fault which originated the abnormal condition (diagnosis stage). This task is complicated since generally there isn't a simple relation between the appearance of a fault and the evolution that it makes in the supervised process. Different faults can cause similar evolutions; on the other hand, the same fault can cause several types of evolutions. This situation is quite a critical problem for the traditional methods of diagnosis.

The diagnosis systems of faults based on artificial neuronal networks (ANN) have achieved a wide level of development and spreading fundamentally because these structures, with architectures of high parallelism, association and fast answer times, make up a very efficient tool for developing in the field of recognizing patterns [4] [9].
In the present work, a theoretical study is realized, based on practical experiences, of the diagnosis process of faults based on neuronal networks. From this study a model of neural network is obtained and a training method with an important modification starting from standard configurations. The experimental results obtained show the superiority of one of the neuronal models proposed and trained with a variation of a classical backpropagation algorithm.

2. Fault Diagnosis

During the operation of a system under supervision (an equipment, a process, a sector of a plant, a complete plant, etc.), the values assumed by the most important variables of the process are captured by a set of sensors strategically put. When the normal status of the process corresponds to a static status, the variables adopt constant values; yet, due to the own noise of the process, it is considered the status as normal while the value evolution of each variable remains within a predefined interval of the original static value; this interval is called band or stripe of normality.
When a fault occurs, the affected variables evolve following defined paths by the fault. At the time when they abandon their respective normality bands, the diagnosis system detects an abnormality and starts the analysis of the observed paths so as to try to identify the problem that originated them.
So as to illustrate this situation, a study of the simplified process is proposed, having one only variable to supervise (for example X temperature), where in addition four potential faults are established identified as f_1, f_2, f_3 and f_4, such as shown in Fig. 1.a.

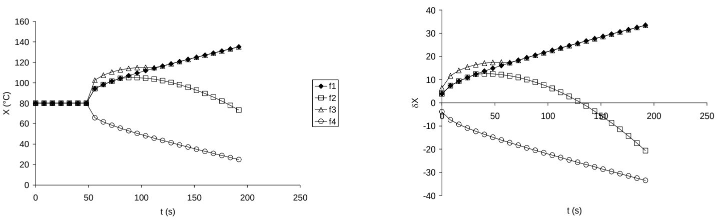


Fig. 1. (a-left) Absolute position regarding path time of X (temperature) for four potential faults. **(b-right)** Typified deviation paths δX transformed by the detector module.

The process operated in a static status, that is to say, while operating normally X adopts a constant predefined value, for example 80ªC. Each time

that one of the faults is seen in the process, the X temperature will stop being constant and will evolve with some of the characteristics paths, where the paths have been generated with a simple interval Δt=8 s.
The proposed diagnosis system of faults and associated to the previous process, uses a set of artificial neural networks (ANNs) specialized in the individual recognizing of each potential fault of the supervised process (Fig. 2). The ANNs, operating in real time and monitoring the evolution of the paths of all and each one of the measured variables, analyze the data coming from the process looking for symptoms or tests for their respective faults.

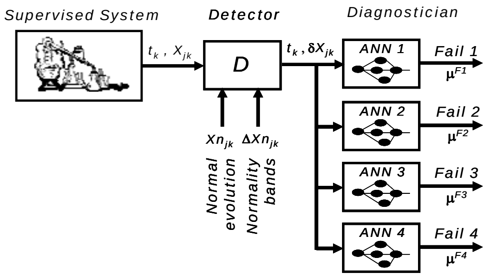


Fig. 2. Diagram of a general system of process-detection-diagnosis

The result of this analysis is the certainty degree $\mu^F \in [0 , 1]$ that each fault supports. When one of the ANNs produces a null value (μ^F=0) it means that all the tests are against the fault that corresponds to it. On the contrary, an equal value to the unit (μ^F=1) implies that there are no proofs against the own fault of the network. Intermediate values represent intermediate supports for the own fault. As can be seen in Fig. 1.a, the variable –temperature X–, remains in its normal value until one of the faults occurs in the activation time t_a=48 s.
Taking as reference fault f_1, at the beginning the f_2 is undistinguishable, but from the t=88s it starts to differentiate. On the other hand, the f_3 path is at the beginning different from f_1, but from t = 112 s it is mixed up with the reference fault. Finally, the f_4 path is in every moment different from the rest of the faults. The two first cases raise a great difficulty to the diagnosis system because different faults originate paths which at some time they are identical. To overcome this problem an optimized method for training the ANNs was developed.

2.1 The Detector Module

For the example under study, a detector module has been used [7] as it is shown in Fig. 2, which converts the absolute paths of X (Fig. 1.a) in the typified deviation paths δX (Fig. 1.b).
In order to comply with its function, the detector gets a data acquisition system –in each sample interval Δt–, the value of X; being the sample k taken in time t_k=k·Δt. On the other hand, the detector also knows the normal value Xn and uses it to calculate the quantitative deviation ΔX=X-Xn. Alter the

quantitative deviation is calculated, the detector uses the normal band ΔXn, which is also known, to calculate the typified deviation δX, as follows:

$$\delta X_k = \frac{\Delta X_k}{\Delta Xn} . \tag{1}$$

Fig. 1.b shows the transformation of absolute paths of X in the typified deviation paths δX, where the regions that present data ambiguity are observed and which must be solved by the diagnosis system.

2.2 The Diagnosis Module

In the example analyzed, the diagnosis system estimates the degree of certainty $\mu_f^F(k)$ which supports fault f in time t_k with the following recursive formulae:

$$\begin{aligned} \mu_f^F(k) &= \mu_f^F(k-1)\ \mu_f^X(k) . \\ \mu_f^F(-1) &= 1 . \end{aligned} \tag{2}$$

where μ_f^X where μx represents the certainty which supports variable X supposing that fault f occurs, as is calculated as:

$$\mu_f^X = \mathrm{fd}\left(\delta X_1, \delta X_f^0\right) . \tag{3}$$

being fd(δX, δX^0) the evaluation function which is used to evaluate the difference between the observed value δX, originated by an unknown fault, and the expected value δX^0 [7] [8].

Considering that ANN1 (Fig. 2) has been specialized in recognizing fault f_1 and observing the exit of this network, Fig. 3 represents the paths of the μ^F that would generate the first neuronal network, –already trained–, of the diagnosis system, while comparing the paths δX of Fig. 1.b, originated by each potential fault of the process, with which it would originate if f_1 were the fault that is really happening. The rest of the networks would produce similar answers when getting data of their respective faults of specialization.

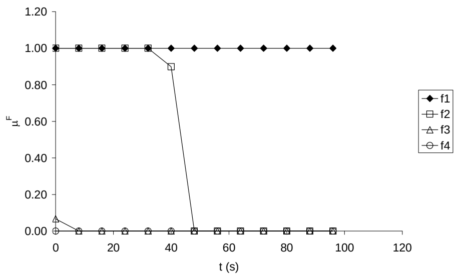


Fig. 3. Paths of μ^F of the ANN1 that supports f_1 facing each potential fault

As can be seen in Fig. 3, when to the first neuronal network (ANN1) enters the path originated by f_1, μ^F it is kept in 1 backing up at every moment to such fault. On the other hand, when the path originated by f_2 is entered, the system does not rule out that it be f_1 since the paths of both faults are similar up to t=32 s. From t=40 s, the system rules out that fault f_1 is decreasing μ^F. When all the paths of f_3 or f_4 are presented, the ANN1 immediately recognizes that they do not correspond to its specialization fault, and fault f_1 is left aside very early.

3. Neural Networks

The artificial neural networks (ANN) can be considered as mathematical models representatives of the brain activity, with the capacity of learning, memorizing and generalizing the learnt information under a diagram of high tolerance to noise, which makes them powerful and versatile tools for the processing of basically numeric information [1] [2].
In this work, three different architectures of neural networks are analyzed: the classical feedforward architecture (Fig. 4) [2] [5], a feedforward architecture with delay windows (Fig. 6) [2] [5] and a feedback or recurrent architecture (Fig. 8) [2] [6], based on equation (2). In addition, two supervised training/learning methods are applied, the traditional method and an optimized method –both based on the backpropagation algorithm–, with the aim of determining the configuration and learning method which are more efficient to solve the raised problem: to recognize the patterns that allow the identification of the fault that gave them rise.
The difference between the two training methods is deep and determining. In the traditional method the output value μ^F for training is set up according to the fault that generated the δX path which is fed as an input to each ANN; that is, μ^F must assume 1 when it enters the δX path of the fault that was assigned to ANN which is being trained, and must assume 0 when it enters the δX path of a different fault, without considering the shape of this path with the rest of them. The latter constitutes the main weakness of the method, originated in the ambiguity of the information that the coinciding segments of the paths δX give caused by different faults (Fig. 1.b), as is the case of the example under analysis. This situation causes the network to try to adjust to contradictory orders.
For the example, in the training of ANN1, specialized in fault f_1, when the δX path is fed generated by f_1, it is taught to make an output $\mu^F=1$; on the contrary when the path δX generated by f_2, ANN is taught to produce an output $\mu^F=0$; yet, since the first 32 s of both paths are indistinguishable, the network will not be able to decide if μ^F must be 0 or 1 during the time when both paths coincide.
Such said problem is solved applying an optimization variant to the training algorithm, when the μ^F value is set according to the coincidence between the observed δX path and the expected path for the network fault and not in the label of the path that corresponds to each fault, such as is done by the traditional method. For the example which is being studied, when the δX path is fed made

by f_1, ANN1 is taught to have an output $\mu^F=1$ because the observed path is the same to the expected one; when the δX path is fed made by f_2, ANN1 is taught to have an output $\mu^F=1$ during the first 32 s since during that time the observed path is equal to the expected; but from that moment, the observed path starts to be different to the expected, and the μ^F value shows this fact decreasing proportionally its value, as shown in Fig. 3. In this way, the ANN1 has no conflict during the learning since for similar inputs δX they must have similar outputs.

Three ANNs models are presented configured with similar architectures and equivalent training conditions so as to be able to compare the results. In relation to practical experiences, only the results of the specialized ANN1 network in recognizing fault f_1 are presented, since it is the one that supports the most unfavorable conditions because of the special configuration of the fault paths.

In relation to the specific training data, each fault path has 13 samples, generated by simulation at intervals of $\Delta t=8$ s (Fig. 1.b). Then, the sequences were concatenated of the four faults, configuring a general sequence of 52 samples.

So as to make the experimental check up the software MatLab® R.9 was used, executed on PC Core 2 Duo equipment with 2 GB of RAM memory. All the ANNs models described in this work have been configured with the neural networks toolbox assistance, available in the software.

3.1 Classical feedforward ANN

It is the typical model of neuronal network used widely for general processes of pattern association. The architecture of each ANN is defined based on the data to be processed and considering the criterion of using an acceptable minimum structure (Fig. 4). According to the Universal Theorem of Function Approximation [2], just one hidden layer is enough for a uniform approximation given a set of training. Due to the dimension of the input and output data, one neuron (fictitious) is used in the input and one in the output. The definition of amount of hidden neurons was experimentally established in four units.

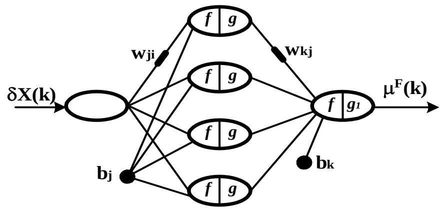


Fig. 4. Classical feedforward ANN architecture

The exciting or net signal f_j of each neuron is defined –according to a classical sketch–, by the pondered composition of the input signals to each processing unit, i.e.:

$$f_j = \sum_{i=1}^{N} w_{ji}\, x_i - b_j\,. \tag{4}$$

where x_i is the output of neuron i of the previous layer made up by N neurons, w_{ji} is the weight of the connection between the present neuron j and the one given by signal i, and b_j is the adjustment weight (bias) of neuron j. For the output signal x_j of each hidden neuron the bivalued sigmoid function was adopted, and the positive sigmoid function for the neuron of the output layer.

3.1.1 Experimental data

The behavior of the ANN1 for identifying the paths of Fig. 1.b was quite unfavorable, both when trained with the traditional method and when they were trained with the optimized method (Fig. 5.a and 5.b).

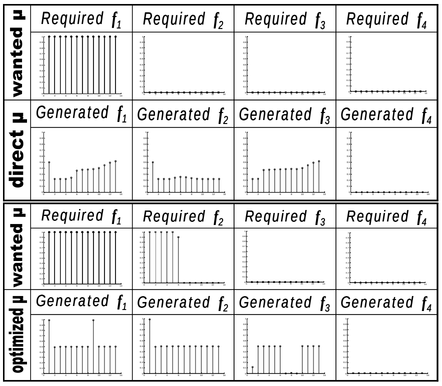


Fig. 5. (a-left) Required/generated output of the classical feedforward model of ANN1. Training: backpropagation standard. **(b-right)** Required/generated output of the classical feedforward model of ANN1. Training: optimized backpropagation.

The training result analysis of the ANN1 network of the diagnosis system – which was trained for recognizing f_1–, reported an μ^F near to 0,5 for the first three faults with both training methods, and a μ^F=0 for the fault f_4. This behavior is incorrect but coherent, since the first three faults present ambiguities which the neuronal network cannot solve, while the fault f_4, since it presents a totally different path than the previous, has been well solved.
The inability of learning of this network is founded in its own structure. In effect, when having to decide the μ^F value (output) according to a unique δX value (input), the ANN1 does not have the necessary information for discriminating among the several paths in which the coinciding segments commit. For example, the input value δX=10 is achieved at some moment by the f_1, f_2 and f_3 paths; and both training methods demand at some moment

output values μ^F different for such input, which confuses the network. One way of solving this problem is to increase the amount of available information for the ANN; two alternatives are explored in this way below.

3.2 ANN with delay windows

For certain dynamic processes, having the information of the late past can improve the behavior of the system [3]. In our case, to increase the information sent to the ANN1, the architecture presented in the previous section was widened through the adding of two additional inputs. These new inputs are obtained by keeping, through delays, the last two observed values of δX, adding two temporal windows (Fig. 6).

3.2.1 Experimental data

At first, this differed time architecture is more appropriate than the direct feedforward; however, its performance –although a little better than the previous model–, has not been satisfactory.
Considering once again the first network of the diagnostician (ANN1) –which was trained for recognizing f_1–, generated some incorrect samples in recognizing its specialization fault with both training methods, but we can see, from Fig. 7.a and 7.b, a better performance due to the delay windows which give more information of the paths.

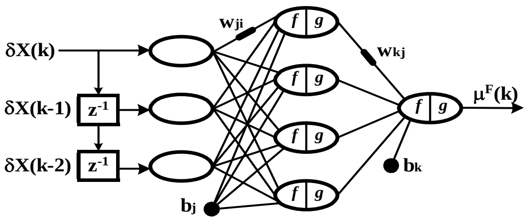


Fig. 6. Feedforward ANN architecture of differed time with two delay windows

The f_2 fault was better solved with optimized training method and with a bad result with the traditional method, because of contradictory data in the first ambiguous area (Fig. 1.b). The f_3 fault was also recognized with mistakes and the f_4 was well solved as in the previous case.

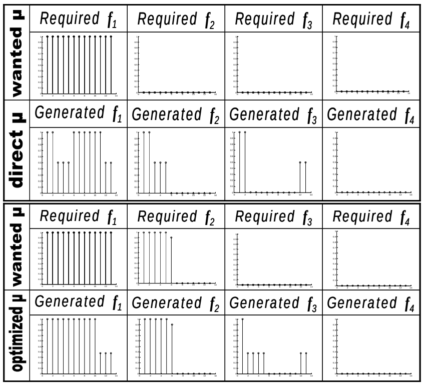


Fig. 7. (a-left) Required/generated output of the feedforward model with windows of the ANN1. Training: backpropagation standard. **(b-right)** Required/generated output of the feedforward model with windows of the ANN1. Training; backpropagation optimized.

The poor behavior with the traditional method is due to its intrinsic weakness because of the overlapping of some segments of the δX paths, but the failure with the optimized method shows a limitation of the proposed structure. This limitation could be founded in the reduced amount of neurons of the inner layer or for the limited width of the temporal window. Yet, here the analysis of this type of network is halted in favor of keeping a similar structure for comparing the networks and while considering the architecture which is explained in the following section.

3.3 Recurrent ANN

The network shown in Fig. 8 corresponds with the Jordan recurrent model which back feeds the output towards a contextual layer of the input [2]. This structure is based in the equation (2), which gives the theoretical basis, and which was deducted when modeling the desired behavior for the proposed diagnosis system [7] [8].
When having a feedback of the output towards the input, the information which the ANN gets, combines the present status and the total history of the process through the successive product of the feedback outputs [6]. It may be considered that the potentiality of the optimized training method is based on this strategy; once the system has detected that the sequence does not correspond to a fault in consideration, it begins to feedback values of μ^F each time minor until they are cancelled and kept in that status, even when the sequence puts again ambiguous values (such as the case in the second ambiguous area of Fig. 1.b).

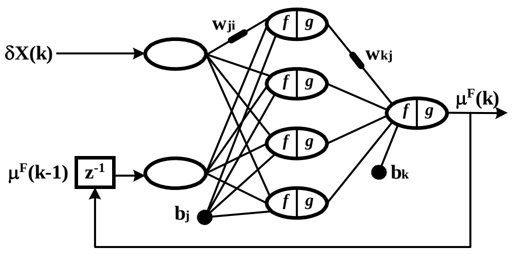


Fig. 8. Recurrent ANN architecture with a delay

The feedback model of Jordan has a great advantage on the non recurrent architectures that can only see a part of the history of the process; the classical feedforward ANN only has the present simple and the differed time ANN just takes two samples of the inputs backwards. The recurrent model used keeps all the history of the process through a unique numerical additional input: the feedback μ^F. The latter is very important when working with a big number of variables.

3.3.1 Experimental data

To the structural simplicity of the recurrent ANN it is added an excellent behavior of the tests made with the optimized method of training. In effect, the ANN1 -corresponding to fault f_1–, trained with the optimized method, was able to generate almost exactly the desired μ^F outputs, truly reproducing the curves of Fig. 3; yet, its behavior was not consistent when it was trained with the traditional method due to the failures of the used method and not of the proposed structure, as can be seen in Fig. 9.a and 9.b.

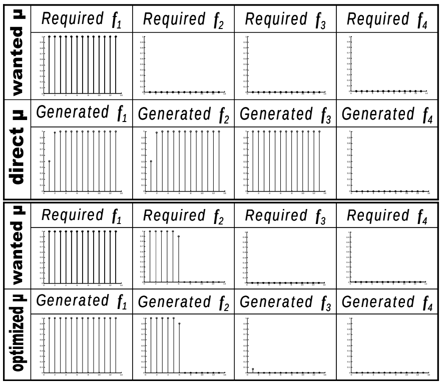


Fig. 9. (a-left) Required/generated output of the recurrent model of the ANN1. Training: backpropagation standard. **(b-right)** Required/generated output of the recurrent model of the ANN1. Training: backpropagation optimized.

The three models of artificial neural networks have experimentally shown their abilities in the recognizing process of patterns. For a better comparison, all the obtained results are briefly shown in the comparison chart of table 1.

Table 1. Comparison chart of the behavior of three neural models and two training methods in the process of recognizing faults

Models → Parameters ↓	Standard FF ANN		Delayed FF ANN		Recurrent ANN	
	direct μ	optim. μ	direct μ	optim. μ	direct μ	optim. μ
MSE of training	$1{,}5x10^{-1}$	$1{,}8x10^{-1}$	$4{,}8x10^{-2}$	$3{,}6x10^{-2}$	$9{,}6x10^{-3}$	$2{,}4x10^{-14}$
Non recognized samples	39	37	14	10	27	2
% of recognizing error	75%	71%	27%	19%	52%	3%
Non recognized samples in f_i	13	11	5	3	1	0
Percentage error in fault 1	100%	85%	38%	23%	8%	0%

4. Conclusions

In this development the efficiency for making identification of paths of faults of three neuronal architectures and two training methods were evaluated. From the study it is deduced that a classical feedforward neural network is not able to assume ambiguous knowledge produced by contradictory information in its training stage. On the other hand, feedforward architecture with delay windows, of comparable complexity to the previous, is also inefficient to learn ambiguous patterns in its training stage, under the two proposed training methods: the Standard backpropagation and the optimized backpropagation.
On the other hand, the combination of a recurrent ANN structure, also of comparable complexity to the previous, and an optimized training method has given an excellent behavior in recognizing fault paths that present coinciding or redundant segments. The feedback structure gave to the network enough information for learning and recognizing complex fault patterns.
Starting from this work, two complementary studies can break down so as to complete the activity of a diagnosis system with neural networks. On one hand, the application of multidimensional fault sequence on models based on ARMA architectures. On the other hand, to improve the training method so that the ANN increases its tolerance related to noise, which could be achieved by adding paths with noise in the training stage, or adding normality bands during the learning stage.

References

1. Anderson, J. (2007). Redes Neurales. Alfaomega Grupo Editor, México.
2. Haykin, S. (1998). Neural Networks – A Comprehensive Foundation. Prentice-Hall, Ontario.

3. Jain, L.C., Martin, N.M. (1998). Fusion of Neural Networks, Fuzzy Systems and Genetic Algorithms: Industrial Applications. CRC Press LLC, Florida.
4. Jain, L., Rao Vemuri, V. (1999). Industrial Applications of Neural Networks. CRC Press LLC, USA.
5. Looney, C. (1997). Pattern Recognition Using Neural Networks: Theory and Algorithms for Engineers and Scientists. Oxford University Press, New York.
6. Mandic, D., Chambers, J. (2001). Recurrent Neural Networks for Prediction. John Wiley and Sons Ltd., New York.
7. Tarifa, E., Martínez, S. (2007). Diagnóstico de Fallas con Redes Neuronales. Parte 1: Reconocimiento de Trayectorias. In Ingeniería e Investigación, vol. 27/1, 68-76, Bogotá.
8. Tarifa, E., Martínez, S. (2007). Diagnóstico de fallas con redes neuronales. Parte I1: Reconocimiento de flujos. In Ingeniería e Investigación, vol. 27/2, pp. 65-71, Bogotá.
9. Zhang, J. (2006). Improved on-line Process Fault Diagnosis through Information Fusion in Multiple Neural Networks. In Computers & Chemical Engineering, vol. 30 – issue 13, 558-571, Elsevier Ltd.

Esta publicación se terminó de imprimir
en el mes de octubre de 2011,
en la ciudad de La Plata,
Buenos Aires,
Argentina.