Grigori Sidorov
Arturo Hernández Aguirre
Carlos Alberto Reyes García (Eds.)

LNAI 6438

Advances in Soft Computing

9th Mexican International Conference
on Artificial Intelligence, MICAI 2010
Pachuca, Mexico, November 2010, Proceedings, Part II

2 Part II

MICAI
2010

Springer

Lecture Notes in Artificial Intelligence 6438

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Grigori Sidorov
Arturo Hernández Aguirre
Carlos Alberto Reyes García (Eds.)

Advances in Soft Computing

9th Mexican International Conference
on Artificial Intelligence, MICAI 2010
Pachuca, Mexico, November 8-13, 2010
Proceedings, Part II

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Grigori Sidorov
Instituto Politécnico Nacional
Centro de Investigación en Computación
Av. Juan Dios Batiz, s/n, Zacatenco, 07738 Mexico City, México
E-mail: sidorov@cic.ipn.mx

Arturo Hernández Aguirre
Centro de Investigación en Matemáticas (CIMAT)
Departamento de Ciencias de la Computación, Callejón de Jalisco s/n
Mineral de Valenciana, Guanajuato, 36240, Guanajuato, México
E-mail: artha@cimat.mx

Carlos Alberto Reyes García
Instituto Nacional de Astrofísica, Optica y Electrónica (INAOE)
Coordinación de Ciencias Computacionales, Luis Enrique Erro No. 1
Santa María Tonantzintla, 72840, Puebla, México
E-mail: kargaxxi@inaoep.mx

Library of Congress Control Number: 2010937860

CR Subject Classification (1998): I.2, F.1, H.3, I.4, I.5, H.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN	0302-9743
ISBN-10	3-642-16772-1 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-16772-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

Artificial intelligence (AI) is a branch of computer science that models the human ability of reasoning, usage of human language and organization of knowledge, solving problems and practically all other human intellectual abilities. Usually it is characterized by the application of heuristic methods because in the majority of cases there is no exact solution to this kind of problem.

Soft computing can be viewed as a branch of AI that deals with the problems that explicitly contain incomplete or complex information, or are known to be impossible for direct computation, i.e., these are the same problems as in AI but viewed from the perspective of their computation.

The Mexican International Conference on Artificial Intelligence (MICAI), a yearly international conference series organized by the Mexican Society for Artificial Intelligence (SMIA), is a major international AI forum and the main event in the academic life of the country's growing AI community. In 2010, SMIA celebrated 10 years of activity related to the organization of MICAI as is represented in its slogan "*Ten years on the road with AI*".

MICAI conferences traditionally publish high-quality papers in all areas of artificial intelligence and its applications. The proceedings of the previous MICAI events were also published by Springer in its Lecture Notes in Artificial Intelligence (LNAI) series, vols. 1793, 2313, 2972, 3789, 4293, 4827, 5317, and 5845. Since its foundation in 2000, the conference has been growing in popularity and improving in quality.

This book contains 44 papers that were peer-reviewed by reviewers from the independent international Program Committee. The book is structured into five thematic areas representative of the main current topics of interest for the AI community and their applications related to soft computing:

- Machine learning and pattern recognition
- Automatic learning for natural language processing
- Evolutionary algorithms and other naturally-inspired algorithms
- Hybrid intelligent systems and neural networks
- Fuzzy logic

The other volume that corresponds to MICAI 2010 contains the papers related to other areas of AI:

- Natural language processing
- Robotics, planning and scheduling
- Computer vision and image processing
- Logic and distributed systems
- AI-based medical applications

We are sure that the book will be of interest for researchers in all AI fields, students that are specializing in these topics and for the public in general that pays attention to the recent development of the AI.

MICAI is an international conference both due to the extended geography of its submissions and for the composition of its Program Committee. Below we present the statistics of the papers submitted and accepted at MICAI 2010. We received 301 submissions from 34 countries, from which 82 papers were accepted. So the general **acceptance rate was 27.2%**. Since MICAI is held in Mexico, we received many submissions from this country, but the acceptance rate for these papers was even lower: 24%. In the table below, the papers are counted by authors, e.g., for a paper by two authors from the country X and one author from the country Y, we added two-thirds to X and one-third to Y.

Table 1. Statistics of MICAI 2010 papers by country

Country	Authors	Submitted	Accepted
Argentina	7	4.00	2.00
Benin	1	0.50	0.50
Brazil	33	13.50	3.00
Canada	3	1.50	1.50
Chile	4	2.00	0.00
China	7	2.50	0.00
Colombia	25	16.67	2.67
Cuba	10	6.78	1.95
Czech Republic	2	2.00	2.00
Finland	1	1.00	0.00
France	11	3.23	0.73
Germany	5	3.25	1.00
Greece	2	0.50	0.00
Hungary	1	0.20	0.20
India	3	1.67	0.00
Iran, Islamic Republic of	9	5.00	1.00
Israel	7	3.33	2.67
Italy	3	0.60	0.60
Japan	4	3.50	2.00
Korea, Republic of	11	4.00	2.00

Table 1. (*continued*)

Country	Authors	Submitted	Accepted
Lithuania	2	1.00	0.00
Mexico	384	186.78	45.53
New Zealand	2	0.67	0.00
Pakistan	9	4.75	2.67
Poland	6	4.00	1.00
Russian Federation	3	2.00	1.00
Singapore	2	2.33	0.33
Spain	22	7.08	2.25
Sweden	1	1.00	0.00
Taiwan	1	1.00	0.00
Turkey	2	1.00	1.00
UK	8	2.67	1.00
USA	19	9.98	4.40
Venezuela, Bolivarian Republic of	2	1.00	0.00

We want to thank all the people involved in the organization of this conference. In the first place, these are the authors of the papers published in this book: it is the value of their research work that constitutes the essence of the book. We thank the Track Chairs for their hard work and the Program Committee members and additional reviewers for their great effort reviewing the papers, allowing their selection for the conference.

We would like to express our sincere gratitude to the Universidad Autónoma del Estado de Hidalgo (UAEH), ITESM Campus Pachuca, Universidad Politécnica de Pachuca (UPP), Rectoría of the UAEH headed by Humberto Veras Godoy, Gerardo Sosa Castelán, General Secretary of the UAEH, and Octavio Castillo Acosta, head of the Basic Sciences and Engineering Institute of the UAEH, for their warm hospitality related to MICAI 2010, and for providing the infrastructure for the presentation of the keynote speakers, tutorials and workshops, and for their valuable participation and support in the organization of this conference. Their commitment allowed that the opening ceremony, technical contributory conferences, workshops and tutorials could be held in the main historic building of the UAEH. We also want to thank the Consejo de Ciencia y Tecnología del Estado de Hidalgo for their partial financial support (project FOMIX 2008/97071) and the Oficina de Convenciones y Visitantes of the State of Hidalgo represented by Ms. Lizeth Islas for their valuable effort in organizing the cultural program as well as entertainment activities. We are deeply grateful to the conference staff and to all members of the Local Committee headed by Félix A. Castro Espinoza and Joel Suárez Cansino.

The entire submission, reviewing, and selection process, as well as putting together the proceedings, was supported for free by the EasyChair system (www.easychair.org). We are also grateful to Springer's staff for their help in preparation of this issue.

Grigori Sidorov
Arturo Hernández-Aguirre
Carlos Alberto Reyes-García

Conference Organization

MICAI 2010 was organized by the Mexican Society for Artificial Intelligence (SMIA, Sociedad Mexicana de Inteligencia Artificial) in collaboration with Universidad Autónoma del Estado de Hidalgo (UAEH), Centro de Investigación en Computación del Instituto Politécnico Nacional(CIC-IPN), Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Universidad Nacional Autónoma de México (UNAM), Universidad Autónoma de México (UAM), Instituto Tecnológico de Estudios Superiores de Monterrey (ITESM), and Centro de Investigación en Matemáticas (CIMAT), Mexico.

The MICAI series webpage is www.MICAI.org. The webpage of the Mexican Society for Artificial Intelligence (SMIA) is www.SMIA.org.mx. Contact options and additional information can be found on those webpages.

Conference Committee

General Chair

Carlos Alberto Reyes García

Program Chairs

Grigori Sidorov
Arturo Hernández Aguirre

Workshop Chair

Gustavo Arroyo

Tutorials Chair

Rafael Murrieta

Plenary Talks and Grants Chair

Jesus A. Gonzalez

Financial Chair

Grigori Sidorov

Best Thesis Awards Chair

Miguel Gonzalez

Doctoral Consortium Chairs

Oscar Herrera
Miguel Gonzalez

Promotion Chair

Alejandro Peña

Local Chairs

Felix A. Castro Espinoza
Joel Suárez Cansino

Track Chairs

Track	Track Chair
Natural Language Processing	Sofia N. Galicia-Haro
Machine Learning and Pattern Recognition	Mario Koeppen
Hybrid Intelligent Systems and Neural Networks	Carlos A. Reyes-García
Logic, Reasoning, Ontologies, Knowledge Management, Knowledge-Based Systems, Multi-agent Systems and Distributed AI	Raul Monroy
Data Mining	Jesus A. González
Intelligent Tutoring Systems	Alexander Gelbukh
Evolutionary Algorithms and Other Naturally-Inspired Algorithms	Efrén Mezura-Montes, Guillermo Leguizamón
Computer Vision and Image Processing	Alonso Ramírez-Manzanares
Fuzzy Logic, Uncertainty and Probabilistic Reasoning	Oscar Castillo
Bioinformatics and Medical Applications	Olac Fuentes
Robotics, Planning and Scheduling	Gildardo Sánchez

Program Committee

Luis Aguilar
Ruth Aguilar
Teresa Alarcon
Alfonso Alba
Adel Alimi
Annalisa Appice
Edgar Arce-Santana
Miguel Arias Estrada
Gustavo Arroyo
Serge Autexier
Victor Ayala-Ramirez
Andrew Bagdanov
Sivaji Bandyopadhyay
Maria Lucia Barrón-Estrada

Ildar Batyrshin
Bettina Berendt
Igor Bolshakov
Ramon Brena
Peter Brusilovsky
Phillip Burrell
Pedro Cabalar
Leticia Cagnina
Felix Calderon
Hiram Calvo
Nicoletta Calzolari
Sergio Daniel Cano Ortiz
Gustavo Carneiro
Juan Martín Carpio Valadez
Jesus Ariel Carrasco-Ochoa
Oscar Castillo
Juan Castro
Mario Chacon
Aravindan Chandrabose
Chuan-Yu Chang
Edgar Chavez
ZheChen
Yueh-Hong Chen
Simon Colton
Quim Comas
Diane Cook
Oscar Cordon
Juan-Francisco Corona
Ulises Cortes
Nareli Cruz-Cortés
Nicandro Cruz-Ramirez
Vicente Cubells Nonell
Alfredo Cuzzocrea
Oscar Dalmau
Justin Dauwels
Jorge de la Calleja
Marina De Vos
Louise Dennis
Juergen Dix
Lucas Dixon
Bernabe Dorronsoro
Beatrice Duval
Susana Esquivel
Marc Esteva
Claudia Esteves
Julio Estrada
Gibran Etcheverry
Eugene Ezin
Luis Falcón
Francesc J. Ferri
Juan J. Flores
Andrea Formisano
Olac Fuentes
Sofia N. Galicia-Haro
Jean Gao
René Arnulfo García-Hernández
Eduardo Garea
Alexander Gelbukh
Fernando Gomez
Pilar Gómez-Gil
Eduardo Gomez-Ramirez
Jesus A. Gonzalez
Arturo Gonzalez
Miguel Gonzalez-Mendoza
Alejandro Guerra-Hernández
Steven Gutstein
Hartmut Haehnel
Hyoil Han
Jin-Kao Hao
Yasunari Harada
Pitoyo Hartono
Rogelio Hasimoto
Jean-Bernard Hayet
Sergio Hernandez
Arturo Hernández
Hugo Hidalgo
Larry Holder
Joel Huegel
Marc-Philippe Huget
Seth Hutchinson
Dieter Hutter
Pablo H. Ibarguengoytia
Héctor Jiménez Salazar
Moa Johansson
Young Hoon Joo
Chia-Feng Juang
Vicente Julian
Hiroharu Kawanaka
Mario Koeppen
Mark Kon
Vladik Kreinovich
Ricardo Landa-Becerra
Reinhard Langmann
Yulia Ledeneva

Additional Reviewers

Table of Contents – Part II

Table of Contents – Part I

Discovering Role of Linguistic Geometry

Boris Stilman[1,2], Vladimir Yakhnis[2], and Oleg Umanskiy[1,2]

[1] University of Colorado Denver, Department of Computer Science and Engineering, Campus Box 109, Denver, CO 80217-3364, USA
[2] STILMAN Advanced Strategies Denver, 1623 Blake Street, Suite 200, Denver, CO 20202, USA
{boris,vlad,oleg}@stilman-strategies.com

Abstract. Linguistic Geometry (LG) is a type of game theory for extensive discrete games scalable to the level of real life defense systems. LG was developed by generalizing experiences of the advanced chess players. In this paper we summarize experiences of highly successful application of LG to a diverse set of board games and military operations. We believe that LG has a more fundamental nature than simply yet another mathematical theory of efficient wargaming. Every LG application generated new ideas that experts evaluated as brilliant. We suggest that LG is a mathematical model of human thinking about armed conflict, a mental reality that existed for thousands of years. The game of chess was invented 1.5-2 thousand years ago (following different accounts) as a formal gaming model of ancient wars. In our case, chess served as a means for discovering human methodology of efficient warfare. To test this hypothesis we would have to demonstrate power of LG software on wars happened at times when the game of chess had been unknown. In this paper, we present an approach to LG-based analysis of the battles of Alexander the Great demonstrating that after tuning the LG-based software will generate the same courses of action as those reported by the historians.

Keywords: Linguistic Geometry; game theory; search; Artificial Intelligence, ancient warfare.

1 Introduction

Linguistic Geometry (LG) is a game-theoretic approach that has demonstrated a significant increase in size of problems solvable in real time (or near real time). The word “Linguistic” refers to the model of strategies formalized as a hierarchy of formal languages. The word “Geometry” refers to the geometry of the abstract game board as well as the abstract relations defining the movements and other actions of the game pieces as well as their mutual influence. It is a viable approach for solving board games such as the game of chess as well as practical problems such as mission planning and battle management. Historically, LG was developed, beginning from 1972, by generalizing experiences of the most advanced chess players including World Chess Champions and grandmasters [1], [24], [32]. In the 70s and 80s this generalization resulted in the development of computer chess program PIONEER utilized successfully for solving chess endgames and complex chess positions with a number of variations considered in the order of 10^2 [4], [32]. These variations were very close to those considered by the advanced chess experts when analyzing the same problems. Further generalization led to development of the new type of game theory – LG – changing the paradigm for solving game problems: "From Search to Construction" [24] – [50], [56], [57]. An LG-based technology was applied to more than 30 real life defense related projects [19]. On multiple experiments, LG successfully demonstrated the ability to solve extremely complex modern military scenarios in real time. Moreover, applications of LG demonstrated the ability of dynamic real-time regeneration of the plans and courses of actions during the simulated combat whenever new information was available. The efficacy and sophistication of the courses of action developed by the LG tools exceeded consistently those developed by the commanders and staff members [49], [50].

Thirty eight years of development of LG including numerous successful applications to board games and, most importantly, to a highly diverse set of modern military operations, from cruise missiles to military operations in urban terrain to ballistic missile defense to naval engagements, led us to believe that LG is something more fundamental than simply yet another mathematical model of efficient wargaming.

At the beginning of a new domain development, the LG construction set – the set of types of zones – is usually expanded to realize the domain specifics employing various experts' ideas. This development quickly saturates itself: usually, several new types of zones permit fully reflect the specifics of the domain. From that moment on, the LG application, itself, begins generating strategies including "new ideas" that experts evaluate as extremely bright and even brilliant. We would like to suggest that LG is a mathematical model of human thinking about conflict resolution, a warfighting model at the level of superintelligence. More precisely, LG is a mathematical model of the brain models [9] utilized by humans for the armed conflict resolution.

To explain chess-related heritage of LG, we should recall that the game of chess was originally invented 1.5 - 2 thousand years ago as a gaming model of ancient wars. Thus, in our case, it served as a means for discovering human methodology of efficient warfare. In the upcoming experiments this hypothesis will be tested by demonstrating power of LG on ancient wars happened at times when the game of chess had not been invented. To set up these experiments we would have to extend the domain of applicability of LG. To discover role of LG in human culture, in this paper, we investigate various issues of applicability of LG to the ancient warfare and specifically to the battles of Alexander the Great, from 334 BC to 331 BC.

2 Applying LG

LG may be structured into two layers: game construction and game solving. Construction involves a hypergame approach based on a hierarchy of Abstract Board Games (ABG), Section 4. Game solving includes both resource allocation for generating an advantageous initial game state and dynamic strategy generation for reaching a desirable final game state in the course of the game. A typical application to a new domain is developed as follows.

First, the problem is defined as an LG hypergame [39], [43], [42], a hierarchical system of several ABG, i.e., the players, the boards, the pieces, the game rules, etc., are identified. Then, LG is utilized to generate strategies guiding the behavior of the players so that their goals would be fulfilled.

Employing LG, the best strategies in ABG are encoded analogously to the genetic code [15]. Indeed, in biology, the genes can be represented as strings of words or triplets consisted of 3 symbols each. Sometimes, triplets are called codons. Each codon contains the genetic code for a single amino acid. For example, the codon Adenine-Uracil-Adenine (AUA) encodes the amino acid Isoleucine. The structure of codons is based on the genetic alphabet of 4 letters representing 4 nucleotides, specialized molecules that — in certain orders, code for the production of proteins. Based on simple combinatorics, it is clear that only 64 (43) different codons are possible, though the nature utilized only 20 standard amino acids. In LG, the alphabet consists of several types of trajectories, i.e., planning sequences of steps [32], [24], [32]. These are several types of the main trajectories such as attack trajectory, relocation trajectory, domination trajectory, etc., as well as negation trajectories of various degrees, first negation, second negation, etc. The actual code in LG, "the set of codons", consists of several networks of trajectories called LG zones. They are attack zone, unblock zone, zone with pared trajectories, zone with restricted areas, etc. (Fig. 1 and [32], [50]). One of the major differences of the LG code is that a zone is not reducible to the linear sequence of trajectories as in genetic code where a codon is a chain of nucleotides, i.e., a 1-dimensional structure. We should keep in mind that while a codon is a complex organic molecule, a 3-dimensional structure, a code represented by a chain of these molecules is a 1-dimensional string of symbols. An LG zone is a network of trajectories, thus, it is, at least, 2-dimensional. Consequently, the grammars utilized in LG to generate the LG code, the language of zones, must be different from the Chomsky grammars [11], which are used to generate and parse linear structures such as natural language, programming languages and the genetic code. To handle 2-dimensional LG code we utilize the Controlled grammars [32], which include powerful tools for managing semantics of n-dimensional space.

The entire strategy includes only actions encoded in the trajectories of zones such as movements along these trajectories, applications of weapons, etc. Like in genetics, where the entire diversity of life is reduced to combinations of 4 nucleotides and 64 codons that encode 20 amino acids, the diversity of strategies in LG is reduced to combinations of several types of zones. This leads to a dramatic reduction of complexity. The strategies are not searched but constructed.

Consider informally a complete set of different zones for a serial ABG such as the game of chess. Formal definitions of these zones are given in [32]. This set includes just five different types of zones: attack, block/relocation, domination, retreat and unblock. These five types represent a complete set of "amino-acids" of the game of chess. Examples of such zones are shown in Fig. 1. For the attack zone, the attack side (white pieces p_o and p_1) is intended to destroy the target q_1 while the intercept side, q_1, q_2, and q_3, is intended to protect it. For the block zone the attack side is intended to block the trajectory of q_1 by relocating po to point 4, while the intercept side is intended to prevent this relocation. This zone is linked to the attack zone of the piece q_1. In general, for a relocation zone, p_o is intended to occupy point 4, but the purpose of that might vary from block to other types of participation in an attack zone.

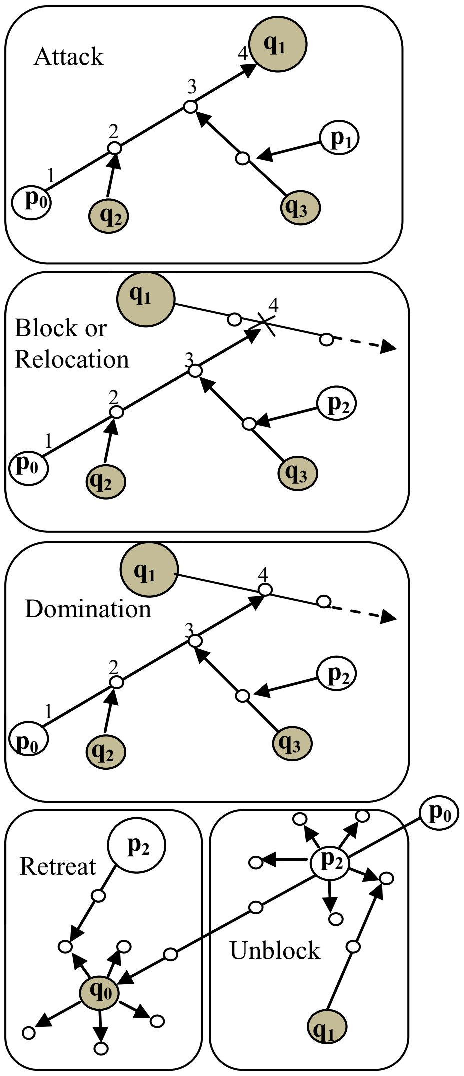


Fig. 1. Complete set of serial LG zones

For the domination zone, the attack side is intended to intercept q_1 at point 4 by dominating this point from point 3 (employing relocation of p_o), while the intercept side is intended to prevent this domination. This zone is linked to the attack zone of q_1. For the retreat zone, the retreat side that includes q_o is intended to save q_o from the attack of p_o by moving it away from the destination of the trajectory of p_o; the intercept side that includes p_1 is intended to prevent this retreat. For the unblock zone, the unblock side is intended to support the attack of p_o along its trajectory by moving the blocking piece p_2 away, while the intercept side (piece q_1) is intended to prevent this unblock. Both zones, retreat and unblock, are linked to the attack zone with main piece p_o.

A set of zones generated in every state of a problem is a unique representation of this state. A piece may be involved in several zones and in several trajectories in the same zone. All the trajectories and zones are evaluated with respect to their quality [32]. Only the highest quality trajectories are considered for generating strategies. The quality function is based on the prediction of the rate of difficulty for a piece for moving along the trajectory. For example, for the attack zone (Fig. 1) piece p_o has to pass three locations 2, 3, and 4 to reach destination and destroy its target at 4. This passage may be free or it may be abstracted by the enemy pieces. For example, piece p_o can be captured at location 2 by q_2. The notion of passage through location for the game of chess is based on the values of pieces (surrounding this location) and on the result of optimal exchange of these pieces [32]. For the military operations employing trajectories of physical entities (missiles, planes, single soldiers) and shooting, the notion of passage through location is based on the notion of probability of kill, which is defined for all the entity-weapon pairs. These probabilities permit calculating qualities of trajectories and zones based on the integrated probabilities of successful passage. For the operations employing trajectories of pieces representing groups of entities that require close encounter with hand-to-hand fighting (like ancient cavalry or infantry) or massive shooting with low probability of kill of each shot (like ancient archers), the notion of passage is based on the notion of attrition rate, a statistical outcome of the skirmish, which is defined for all the pairs of antagonistic groups (Section 3). These attrition rates permit calculating qualities of trajectories and zones based on the integrated attrition resulting from the passage of these trajectories and zones, respectively. In all cases, the less "difficulties" a piece would experience in passing along a trajectory the higher quality of this trajectory is. Every location along a trajectory, where a piece can be intercepted (for the game of chess), destroyed with high probability (for modern military operations) or suffer high attrition (for ancient operations) reduces quality of this trajectory. A trajectory which includes at least one such location is called a trajectory with closed location or closed trajectory. A trajectory without such locations is called an open trajectory. Consider an open main trajectory of an attack zone (like trajectory for p_o, Fig. 1, top) and a retreat zone for its target (such as q_1). Assume that all the locations for retreat are closed, then the target is called vulnerable and the main trajectory in the attack Zone is called a trajectory with a vulnerable target.

This approach permits to introduce function of quality based on the ordering of the main trajectories and the respective zones beginning from the trajectories with vulnerable target at the top of the list, then open (non-vulnerable), then with one closed location, two, and more. A similar ordering, with the exception of vulnerability, can be done to any trajectory of a zone [32].

3 "Discrete Universe" of Ancient Battles

All the existing applications of LG are based on the software tool kit, LG-PACKAGE, which includes 6 software tools: GDK (Game Development Kit), GRT (Game Resource Tool), GST (Game Solving Tool), GIK (Game Integration Tool), GNS (Game Network Services) and GMI (Game Mobile Interface) [19].

GDK permits to define a “discrete universe” of ABGs by observing “the laws of discrete physics” [32]. The problems in such universe look similar to the generalized board games like chess, checkers, etc. An abstract board, an area of the discrete universe, is represented by an arbitrary finite set. Abstract pieces represent the agents standing or moving with a constant or variable speed. GDK permits to introduce concurrent movement of multiple pieces, application of weapons, communication delays, skirmishes of agents (with proper attrition), collisions, etc.

To apply LG to ancient wars we will introduce new classes of ABGs and LG hypergames representing Assyrian and Persian Wars, Classic Greek Wars, Ancient Roman Warfare, etc. This introduction will include:

- optional cellular structure of abstract boards, which would permit representing various types of terrain, dense and sparse military formations, strategic and tactical maneuvers, etc.; for the Alexander the Great battles we will choose a board of
  - “hexes”, rectangular hexagonal prisms of 30m across the foundation and 3m height (Fig. 2);

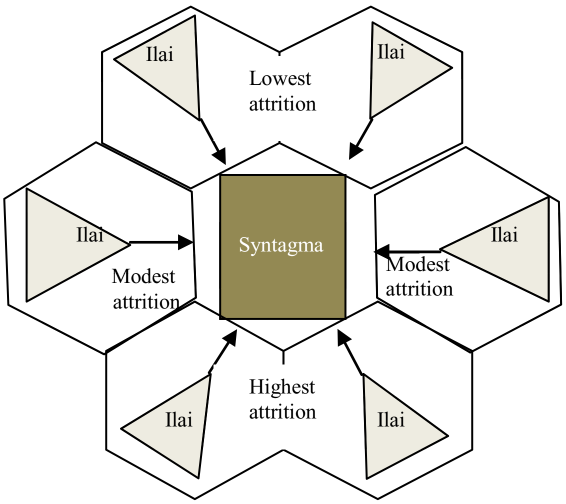


Fig. 2. Rate of attrition for attacks of a Macedonian cavalry piece (ilai) against a Persian piece of phalangites (syntagma)

- abstract pieces representing singular fighters, small groups and subgroups such as Greek Enomotia or Roman Manipula, larger groups such as Greek Lochos (battalion), Roman Cohort and Legion, etc.; for the Alexander battles we will introduce generic pieces representing various units of infantry and cavalry, e.g.,
  - a *syntagma*, a squadron in the Macedonian phalanx, which consisted of multiple syntagmas of square shape of 16×16 men; phalangites were armed with *sarissa*, a pike of 7m long, 4m of which was projected in front of the first line of phalangites when couched for the charge [1], [54]; this type of formation was almost impenetrable from the front but vulnerable from the sides

  and back (see also Fig. 1 and below); here, in figures, we mark syntagma with a square;
  - ➢ an *ilai*, a squadron in the Macedonian heavy cavalry (Companions), which consisted of multiple ilais of 16×8 men of various shapes (square, rhomboid, upside down wedge, etc.) [1], [54]; here, in figures, we mark ilai with a triangle;
- time interval representing physical time required for the real world system to move between two consecutive states represented in ABG; in Alexander's wars we will consider 30 sec and 1 min intervals;
- movement reachabilities for light and heavy infantry, cavalry, chariots, and other pieces representing advancements of these pieces within one time interval;
- probabilities of kill representing a statistical outcome of attack of one piece, a physical entity, by a weapon; all combinations of piece-weapon are considered;
- attrition rates between each pair of adversarial pieces representing groups, e.g., an ilai against a syntagma (Fig. 1) rates are defined with due respect to the direction of attack; in Fig. 1 rate of attrition of the attack piece complements attrition of the target piece, i.e., lower attrition of the ilai corresponds to the higher attrition of the enemy's syntagma.
- weapon reachabilities representing ancient "firepower" such as archers, hoplite phalanx, skirmishers, slingers, legionaries, etc.;
- communication and logistics constraints, e.g., limited communication between commander and subcommander who may each control a half of the army.

Introduction of the ancient wars into GDK would allow us to immediately begin experiments by applying LG game solving tools. For Alexander's wars we will use two versions of LG, Persian and Macedonian. Each of them will utilize GRT to optimize the initial disposition of the battle (the start state) for both adversaries separately and GST to generate best strategies. Most likely, at the initial stage, the start state and the strategies will be far from the optimum. However, these experiments would allow us to move to the second step of development of the LG application.

The second step includes discovery of the new types of LG zones ("genetic alphabet") that represent new domain, the domain of ancient operations. The zones' discovery cycle has been developed and tested on multiple projects for various modern military operations. At the beginning, new zones are just the existing zones with minor differences deduced originally from conversations with SMEs (Subject Matter Experts - military experts and historians) and historical literature. These differences may include rules (and principles) that are specific to a given problem domain or, even, to a particular problem. Execution of the LG application with new zones will permit constant refinement of those principles. The subsequent development will lead to converting specific rules and principles, initially, to generic principles, and later to the universal conceptual zones. The universal new zones will become components of the theory of LG to be used across a variety of problem domains including all the wars of Antiquity and Middle Ages.

The third step, a series of experiments with the new LG application, will provide SMEs with advanced what-if analysis. It will include courses of action generated by LG under the same constraints as those that limited ancient commanders thousands of years ago. Below, in this paper, we argue that these courses of action will be very close to those happened in these battles. In addition, LG will generate alternative COA that could have taken place if the ancient commander would have made another decision or some random events took a different turn. This information will generate feedback from the SMEs that will support further enhancements to the new application.

4 Ancient Battles: The LG Analysis

Below, we will consider several major battles of Alexander the Great in terms of LG. We will quote extensively from the books describing history of Alexander the Great battles [1], [8], [54], all of which are based on the work by Arrian [2], written more than 1,900 years ago, who himself studied publications written by the witnesses of these battles and, unfortunately, are not available to modern historians. We will try to make sense of these battles through the eyes of LG and explain that there is not much difference, if at all, between this view and conventional military and historical accounts. In addition, we will explain the actual courses of action employing notions of LG. Figures (Fig. 3 - Fig. 5) included in this paper illustrate major moments of these battles by showing a snapshot of the LG representation of the battle for each of those moments. Various shapes shown in figures represent abstract pieces for both sides in a conflict (Fig. 2). Numerous thin lines with arrows (Fig. 3 - Fig. 5) are not simply directions for the troops' movement. These are the subsets of the LG zones for the specific moments of the engagements. For simplicity, we included only main trajectories of the zones. These are mostly attack or retreat zones. In some cases we included also first negation (or intercept) trajectories. Also, for simplicity, all the trajectories in figures are represented by smooth curves instead of segmented lines where each segment represents a single move (compare with Fig. 1). The set of zones is dynamically regenerated after every concurrent move, which takes place during one time interval (in our case, it is 30 sec). Trajectories in LG are the planning sequences of steps, which may become the routes that actually took place in the battle. Otherwise, they may change in the course of the battle before the pieces have reached their destination. It would be impossible to present all the information about the battles encoded in the LG representation employing just 6 pictorial snapshots (Fig. 3 - Fig. 5). Thus, Sections 4 - 7 include extensive comments to these snapshots.

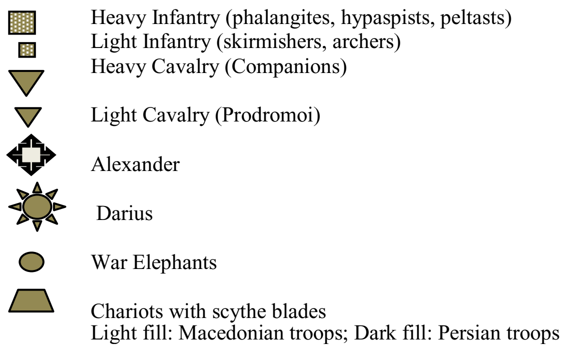

Heavy Infantry (phalangites, hypaspists, peltasts)
Light Infantry (skirmishers, archers)
Heavy Cavalry (Companions)
Light Cavalry (Prodromoi)
Alexander
Darius
War Elephants
Chariots with scythe blades
Light fill: Macedonian troops; Dark fill: Persian troops

Fig. 3. Legend to Figures of Battles

We assume that we have already defined respective ABGs (Section 3). As we discussed in Section 1, in LG, game solving involves two steps, i.e., resource allocation for generating an advantageous initial game state and dynamic strategy generation for reaching a desirable final game state in the course of the game. For the battle planning, this means, at first, to create an opportunity for existence of a winning strategy by optimizing start state of the battle, i.e., optimizing initial resource allocation (choosing the battlefield, the disposition of forces, the time when battle starts, etc.), and then, when the battle begins, to generate and implement optimal strategy leading to victory. In LG, both adversaries do their own start state planning based on their knowledge about the enemy and environment (which is called blue and red worldviews). New information about the enemy may cause reallocation of resources by either side. This preliminary maneuvering (before the battle) converges eventually to the common start state and the battle begins. Obviously, both sides plan for the victory. This means they are trying to achieve their goals with minimal losses. However, only one side is successful. The battle itself makes this decision. The battle shows whose prediction of success was more realistic, whose worldview was closer to reality and whose strategy was closer to the objective optimum. Additionally, the battle reveals who was gravely punished by the enemy for his mistakes and who was able to recover (during the battle) by readjusting his strategy to the changing conditions. The LG tools will do preliminary iterative resource allocation and simulate the battle leading to the final decision. We will show that Alexander the Great and his opponents were, indeed, doing these optimization steps and, most importantly, their reasoning was completely within the scope of the LG tools.

One of the typical resource allocation maneuvers made (or attempted) by Alexander and his opponents was to avoid outflanking by extending his line at the battlefield to match that of the opponent. These equal extensions are well within the scope of the LG strategies. Indeed, if one of the adversaries would manage to outflank the enemy and avoid being outflanked by him, he will be on his way to victory by pursuing a high quality trajectory with vulnerable target (Section 2). In such case the enemy will be hit in the weakest spot (to the rear) and will be surrounded with nowhere to go. This is in part what Alexander achieved in a number of battles, e.g., with 10,000 adversarial Greek mercenaries in the Battle of Granicus (Section 5). Even anticipation of such course of events, i.e., becoming a vulnerable target, would usually cause the enemy to generate retreat zones and withdraw. Outflanking is the type of initial resource allocation intended to create an opportunity of turning the enemy into a vulnerable target in the course of the battle.

Philip, the Alexander's father, had created an integrated army, which included heavy and light infantry (phalanx, hypaspists, peltasts and skirmishers), heavy and light cavalry (Companions, Prodromoi, etc.) and a large body of technicians and engineers (utilized mostly for sieges) [1], [2], [8], [54]. This diversified integrated army allowed Alexander to use different instruments for different tasks. For example, Macedonian phalanx, an impenetrable heavy formation with long (up to 15ft) spears, was not intended to attack enemy phalanx or even win the battle on its own. In LG terms, this means that an attack trajectory of Macedonian syntagma to the front of the enemy's syntagma would be a trajectory of low quality due to high attrition rate for the attack side at the target location. Two mutually adversarial syntagmas could certainly block each other's movement. However, Macedonian phalangites were used to meet and pin down the enemy line (including enemy phalanx and cavalry) while Macedonian cavalry and light infantry penetrated gaps and hit rare. As historians like to underscore [8], [54] the Macedonian phalanx became the anvil against which enemy forces were driven by encircling cavalry, the hammer, and smashed the battle. This means that to resolve a battle Alexander strived to utilize high quality trajectories such as those for Companion cavalry ilai leading to the rear or sides of the enemy's syntagma (Fig. 2) while forcing the enemy retreat following low quality trajectories threatened by the front sides of the Macedonian phalanx. Essentially, these were multiple zones with main trajectories with vulnerable targets (Section 2). Utilizing trajectories with low attrition for his army and forcing the enemy into the high attrition trajectories led Alexander to success with low casualties.

In our analysis of the Alexander's battles, below, we will consider the actions of the ancient generals and interpret them as LG optimization steps, i.e., resource allocation, strategy generation and implementation (via a simulated battle).

5 The Battle of Granicus

For his first battle in Asia Minor, the Battle of Granicus, to avoid outflanking, Alexander extended his line to match that of the Persian cavalry (Fig. 4, A), while the Persian approach to initial resource allocation mystified military historians. The Persians had selected the site at a point where Granicus, a fast flowing and relatively deep river, had steep banks. When Alexander with the vanguard of his army approached the river, his scouts reported that the Persians were holding the river with 20,000 cavalry along their bank. On higher ground behind them, stood 20,000 Greek mercenaries in phalanx formation. The line of Persian cavalry was 2.5 miles long (Fig. 4, A). By placing their cavalry along the banks of the river they made impossible to do what cavalry does best – charge at the gallop, i.e., all the attack trajectories would be crossing the river and, thus, would be of low quality. In positioning their heavy infantry (phalangites) far to the rear (Fig. 4, B), in separate line, they violated the principle of integrated armies, i.e., concurrent attack and intercept trajectories by both cavalry and infantry became impossible. Moreover, this separation permitted Alexander to develop his initial resource allocation employing the strategy with trajectories of unusually high quality. These were combined attack trajectories of infantry and cavalry, first, against a line of Persian cavalry unsupported by infantry, and then against a line of infantry unsupported by cavalry.

Some historians believe that the Persians' initial resource allocation was based on a single "absurd" objective – to kill Alexander. By placing their cavalry along the river they hoped to concentrate on the units led by Alexander and thwart his invasion by killing him. It is more likely that they expected to be outnumbered by the Macedonian infantry and lacked confidence in their own Greek mercenaries. All such hypothesis will be tested in the LG-based resource allocation for the Persian side.

The special cavalry squadron, under the command of Socrates (Fig. 4, C), moved towards the center of the Persian line and charged with some infantry support directly into the midst of the Persian cavalry. This attack was conducted along low quality trajectories with high attrition, thus the attack squadron suffered extremely high casualties.

However, unsupported by infantry Persian cavalry suffered significant losses as well which led to weakness use by Alexander to storm through behind it with cavalry Companions supported by infantry (Fig. 4, D). This secondary attack employing high quality attack trajectories forced a gap in the enemy line, because the Persians began to retreat employing almost randomly chosen retreat trajectories (Fig. 4, D).

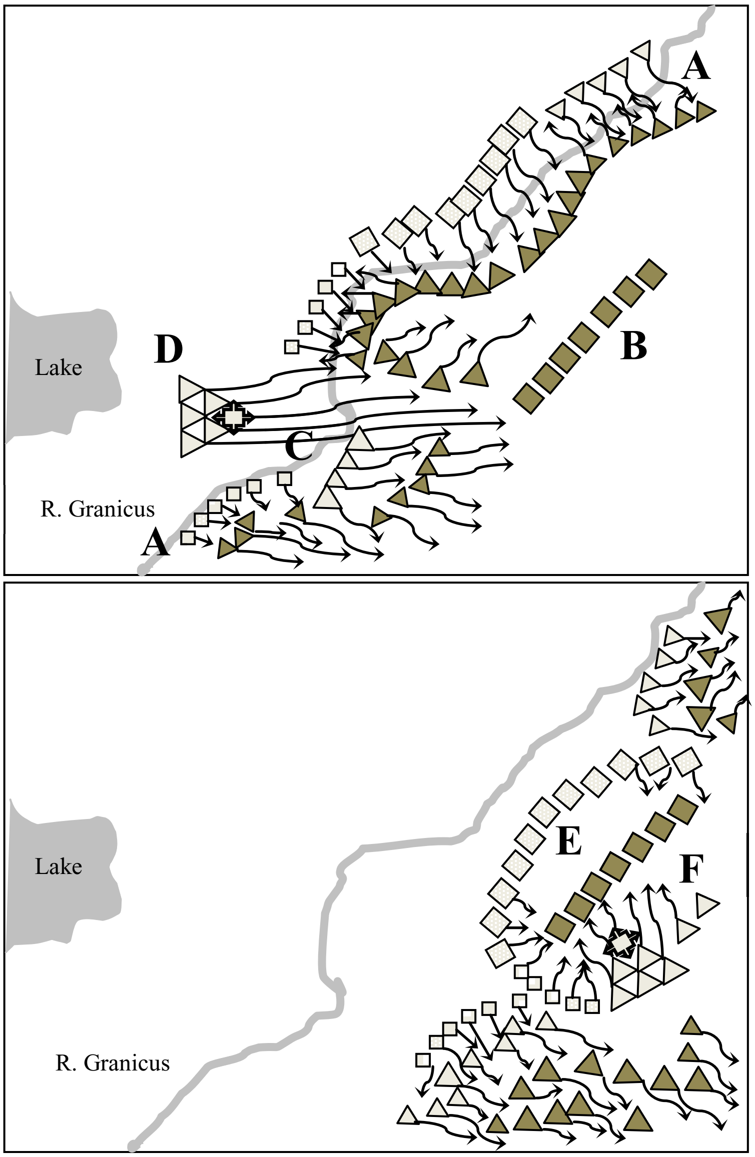


Fig. 4. The Battle of Granicus, 334 BC

When the Persian center broke, the wings fell back too under pressure of the rest of Macedonian cavalry and infantry. The total retreat shown in Fig. 4, bottom, did not include Greek mercenaries – they were holding hopeless line (Fig. 4, E). Alexander did not make a mistake of pursuing the defeated Persian cavalry too far. He could easily reason that pursuit trajectories, while open, were not trajectories with vulnerable targets, i.e., the targets could successfully retreat. Instead, he turned against Greek mercenaries employing the best concurrent attack trajectories of both cavalry and infantry by cutting off the targets retreat. He used his hammer-and-anvil tactics by employing his cavalry to attack mercenaries flanks and rear and his phalangites and light infantry to squeeze them from the front (Fig. 4, F). All the Macedonian trajectories pushing mercenaries to this pocket with nowhere to go were the trajectories of extremely high quality with lowest attrition.

Out of 20,000 mercenaries only 2,000 were taken alive (as slaves). Alexander lost fewer than 150 men during the entire battle, most of them in the initial cavalry assault against the Persian center.

6 The Battle of Issus

Darius, the King of Persia, had placed his army along a riverbed of Pinarus (Fig. 5), which was dry, since it was late Fall of 334 BC. The battleline along the Pinarus extended some 2.5 miles from Mediterranean coast (The Gulf of Issus) up into surrounding hills. Ancient historian Arrian [2] claims that Darius had 600,000 men altogether but no modern historians accept this figures. Even, if the Persian strength was only 200,000 against Alexander's 40,000, his army was greatly outnumbered, but because there was not much level ground he was able to extend a line along Pinarus as long the Persian one. Darius made two resource allocation maneuvers to outflank the Alexander's army at the preliminary stage. The first maneuver was to place some of his men to the high ground on his left across the Pinarus in advance of his line (Fig. 5, A). To counter them, in his resource reallocation, Alexander sent a small force of light troops to drove the Persians higher into the hills and out of the action. The second Darius' maneuver was to concentrate almost all of his cavalry on his right wing on the level ground by the coast (Fig. 5, B). He hoped to generate multiple attack trajectories to crush Alexander's left in a massive charge and to wheel his cavalry around to smash Macedonian infantry in the rear, thus turning the entire Alexander's army into a vulnerable target. When Alexander saw the Persian deployment, as he approached Pinarus, he made another reallocation by strengthening his left wing by sending part of his heavy cavalry units to the left (Fig. 5, C).

As historians underscore [2], [8], it is ironical that Alexander assumed essentially "symmetrical" battle plan and, respectively, the "mutually agreeable" initial disposition as his opponent (for the common battle start state). Alexander hoped to break through Darius' left center in a cavalry charge and to wheel around against the rear of the Persian center in an identical hammer-and-anvil operation. However, while Darius was planning an outflanking trajectory with vulnerable target, Alexander planned similar trajectory by frontal charge through Persian heavy infantry (Fig. 5, D). As we discussed in Section 3 (Fig. 2 and [8]) a cavalry charge directly into a heavy infantry formation cannot be successful because attrition rate is very high. In LG terms, such trajectories are considered to be closed. However, this is the case if the infantry holds and does not panic. This is a probabilistic event and should be accounted as such in computation of the quality of the attack trajectories. Alexander, certainly, gambled but with high probability of breaking the line based on the plan to drive Persian skirmishers back into the infantry ranks, which would demoralize and break the enemy.

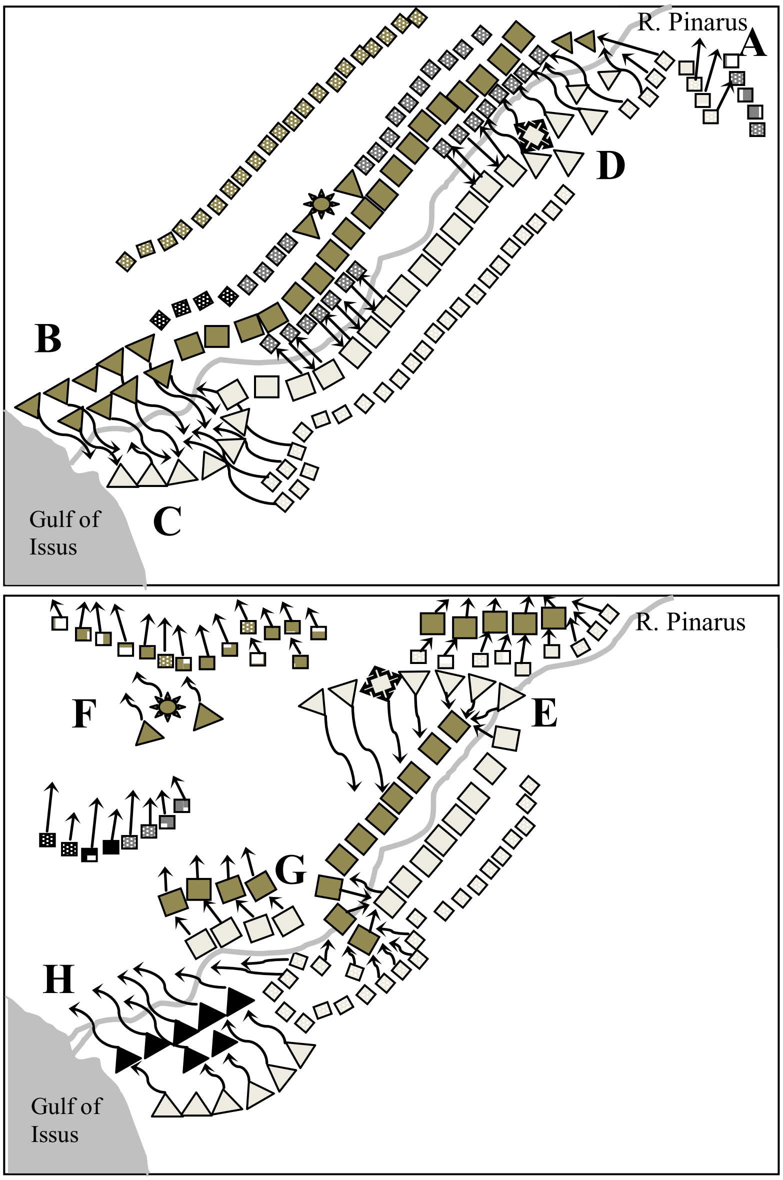


Fig. 5. The Battle of Issus, 333 BC

When Macedonians were within bow shot, at about 100 yards of the Persian line, Alexander ordered attack. With contingents of cavalry on his right, supported by infantry close behind, he charged the river at a gallop and continued frontal attack against Persian phalangites. Despite it was an attack along the closed trajectories, they opened up because Persian light infantry (skirmishers) in front of their line fell back in panic, creating confusion in the main infantry line on the Persian left center, and Alexander smashed through and attacked Persian infantry in the rear (Fig. 5, D and E). This, of course, was a triumph of his initial resource allocation, his strategy development and implementation. Moreover, it shows that Alexander's worldview was correct, very close to reality. At the same time, it shows that Darius resource allocation was imperfect, his strategizing was faulty, and most importantly, his worldview, especially, his knowledge of the strength of Macedonian forces was inaccurate.

When Darius saw that Macedonians wheel behind his lines, he panicked and fled in his chariot (Fig. 5, F). At the same time, Macedonian left wing crossed Pinarus and pressed forward, which resulted in a gap in the Macedonian line. Persian phalangites used this gap, crossed Pinarus and tried to encircle Macedonian phalangites turning them into a vulnerable target (Fig. 5, G). Alexander had to turn against them and lost his chance to pursue the fleeing King of Persia. In the meantime, the Persian right wing cavalry attack was plugged by the Macedonian left wing cavalry units supported by skirmishing light troops; the line was holding. When the Persian cavalry saw that their center was collapsing and that the king had abandoned the field, they too broke and fled (Fig. 5, H). The effect of slow but steady spreading the news that Darius had abandoned the army will be modeled in LG via communication delays between different components of the Persian LG, which will control components of the Persian Forces.

7 The Battle of Gaugamela

Gaugamela, the third and decisive battle of Alexander the Great for conquering Persia, is considered the greatest victory of antiquity. It is especially interesting to investigate if LG tools are adequate to formalizing its course.

As historians underscore the preliminary planning and resource allocation for this battle were substantial although no sources actually describe it in detail. All the decisions and maneuvers before the battle demonstrate that very serious reconnaissance was performed and utilized for preliminary resource allocation and throughout the battle. We will employ the LG tools to realize and explain multiple iterations of mutual maneuvering and resource reallocation for both sides.

In the Spring of 331, Alexander with the army of 47,000 left Egypt for the long march into Persia. When he reached Euphrates in early August, he learned that Darius was waiting for him far to the south with a large army outside Babylon on a site to his advantage. At this time Alexander made a major maneuver. Instead of approaching the goal of invasion and rushing south, down Euphrates, where problems of supply were great, simply to fight Darius on the site of his choice, he crossed the river, turned north for a while before swinging east, keeping the foothills of Armenian mountains on his left in country which offered pasturage for the horses and supplies for the army. In LG terms, instead of pursuing a trajectory directly attacking the target, the

Persian army, Alexander had chosen a "free search" trajectory provoking Darius to intercept him and, thus, move away from the friendly terrain. And so it happened. Darius abandoned his position at Babylon and began moving north across Tigris, which he hoped to use as his line of defense against Alexander. The specific choices of the high quality trajectories for both sides were based on the terrain preferences and are totally within the scope of the LG tools.

When Alexander learned from captured Persian scouts Darius' maneuver and intent, he made another maneuver. He interrupted his "free flight", made a forced march to Tigris, crossed it unopposed, and headed south towards Persian army. In four days he made a contact with the Persian vanguard. This was the plain of Gaugamela, near the town of Arbela. The first stage of resource allocation in choosing a mutually acceptable site was finished.

For four days the Macedonians rested and fortified a camp, so that when time comes Alexander was with a streamlined, mobile striking force. Also, this encampment maneuver within 4 miles from Darius army allowed him to prevent a surprise attack and to observe possible battlefield traps to be prepared by Persians. In his preliminary resource allocation, Darius almost succeeded in selecting his own ground for the battle. Macedonian generals tried to persuade Alexander to immediately attack the Persians as soon as they were spotted but Alexander refused.

He decided to take time for building a camp, inspect grounds between armies and position his forces in sophisticated formations. Alexander led in person a cavalry reconnaissance of the grounds to make sure that there were no cavalry-traps in the form of pits or spikes. In this second stage of resource allocation, Alexander faced a very complex task. The Persian army was enormous, but, probably, less than 1,000,000 as reported by Arrian [2]. Even with a more realistic number 100,000-250,000 of Persian troops against Macedonian 47,000, Alexander was significantly outnumbered in both cavalry and infantry. Additionally, Darius prepared several surprises. The first one was 4 squadrons, each fifty strong, of special chariots equipped with scythe blades upon their wheels and traces, designed to inflict dire casualties upon Alexander's infantry. The second surprise was 15 Indian war elephants. Armed with this information about the enemy, two adversarial versions of LG will complete second stage of resource allocation. It would be extremely educating to learn under which additional constraints, if any, the LG-based resource allocation and strategies would model the decisions of both commanders. As a result we may discover that these additional constraints were actually present at the battlefield but had been missed by the generations of historians.

In his final stage of resource allocation to be simulated by the Persian LG (Fig. 6, A and B), Darius was able to extend Persian line for several miles (Fig. 6, A) with no match from Macedonian side. Darius initial disposition expected double envelopment, a massive cavalry attack around both wings of the Macedonian army, and to terrify the Macedonian infantry in the center with a chariot attack (Fig. 6, B). As we discussed in Section 4, such envelopment means pursuing two sets of attack zones with the same vulnerable target, in this case, the entire Macedonian army.

In his final stage of resource allocation to be simulated by the Macedonian LG (Fig. 6, from C to G), Alexander made several important strategic choices [1], [2], [8], [54] with the main goal to prevent the envelopment, which he predicted. He put his left wing forces in echelon formation to fight holding, defensive battle (Fig. 6, C) similar to the one he had used at Issus (Section 6). Additionally, to avoid Persians flanking him and getting behind his forces, he positioned his crack infantry called hypaspists and his Thessalian allied cavalry to both ends of his line (Fig. 6, D). Both arms were broken into small units, which could move aside in the face of elephants, chariots or cavalry push, showering javelins on those, as he assumed that these units would inevitably be driven back by Persian thrusts. The retreat of various units would become an asset to Alexander's plan, because much deeper in his formation behind front lines he placed reserve phalanx (Fig. 6, E). As his smaller army would wade into the massed Persian forces it would become stronger as driven into itself. Alexander's formation at Gaugamela became a forerunner of a renowned British "square" of the 19th century, which was also getting stronger when it was driven into itself. One of the most interesting and inspiring tactical allocations made by Alexander was his decision of moving elite cavalry and phalanx at the oblique angle into the much longer Persian line (Fig. 6, F). It was a tendency of all shoulders with shields to move to the right in the course of advance, as each man sought to cover the exposed portion of his body with the shield of his comrade next in the line. Theban militants had learned to exploit that tendency with obliquely directed and focused attack on an advancing enemy line as it stretched and thinned in response. Philip, Alexander's father, had spent time in Thebes, and the tactic became a vital part of Macedonian forces. For himself, with Companion cavalry on the right (Fig. 6, G), Alexander was looking for an opportunity to break through the Persian line and to catch some of the Persian forces, possibly, including Darius, between the hammer of the Macedonian cavalry and the anvil of the phalanx. As was the case in his previous battles he planned to turn the enemy into a vulnerable target.

The battle started on October 1, 331 BC. Macedonian left and center oblique formations (multiple phalanx) were edging diagonally right off the area cleared by Darius for his chariot attack (Fig. 6, H). Alexander-led right-wing cavalry was moving still farther to the right. The more numerous and extensive Persian line was mimicking this Macedonian movement by moving correspondingly in the same direction, so it might continue to outreach Alexander on the flank. This way both armies were shifting to the right with Persians extending their line to keep their outflanking option open. With this kind of shift the initial network of LG trajectories and zones will also be shifting to the right but with important caveat. The quality of trajectories and zones will change according to the quality of the new terrain. Both commanders understood this very well. Had this drift persisted, both armies would have slid away from the ground, which had been leveled for the chariots.

Darius decided that overextension of the Persian line caused by the drift of the Macedonian phalanx (which would make chariots useless) had to be stopped. He ordered his left wing cavalry to attack Alexander's right wing. Macedonian right was pushed back by heavy cavalry but the line was holding (Fig. 6, I). At that point Darius signaled the chariot attack (Fig. 6, J). The Macedonians opened their ranks and allowed them to pass through, while troops bombarded them with javelins and arrows, grasped the reins of the horses and dragged down the drivers. Those chariots that got deeper into the open Macedonian ranks were rode down by the cavalry grooms in the rear. The secret weapon had misfired. The zones and the outcome of the chariot attack will be generated by the LG tools based on the attrition rates of the chariot trajectories attacking light infantry formations and cavalry. Why would the Persian LG utilize such trajectories at all? The adversarial casualties and attrition rates for the chariot attack estimated by Persians were incorrect: they were overly optimistic. Based on these attrition rates the Persian LG will misfire as Darius did 2,300 years ago. The rate of attrition expected by Persians would have been confirmed if Darius had employed these chariots for the attack for which they were most suited, i.e., to increase the slaughter of a disorganized and retreating enemy. By the way, elephants had not been used by Persians in this battle because they felt no stake in the battle sufficient to wade into the bristling spears and arrows of the enemy, and proved useless and uncontrollable, more a danger to their own side.

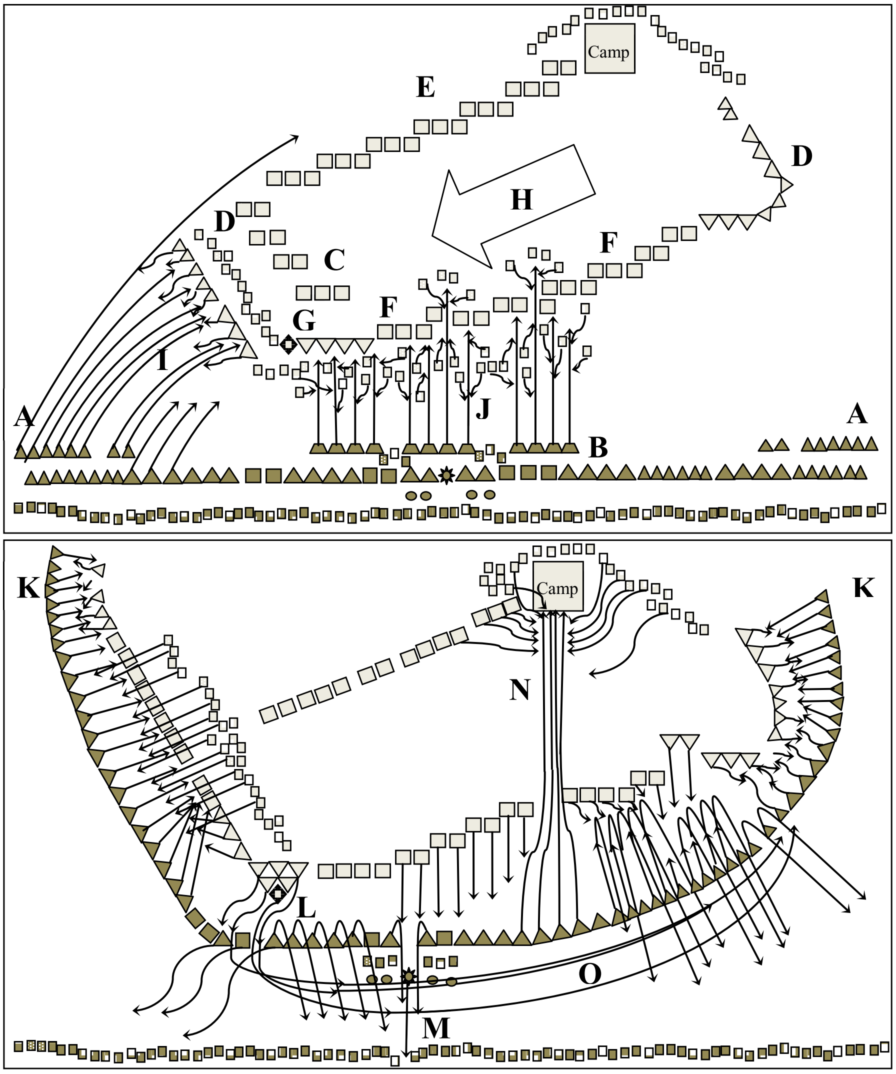


Fig. 6. The Battle of Gaugamela, 331 BC

Darius signaled implementation of his main plan of double flanking and totally encircling the Macedonian army. His left wing cavalry fighting already with Macedonians began extending to the north. His right wing flanking maneuver began with heavy attack on Alexander's left and quickly outflanked it (Fig. 6, K). When Persians were attacking Alexander's right, some of their cavalry units dashed ahead and created a gap between Persian left infantry and cavalry. Alexander saw this gap and immediately wheeled to charge it in a wedge cavalry formation of his Companions (Fig. 6, L). Simultaneously, the Macedonian phalanx charged frontally at the center of Persians. As Alexander with Companions broke through Persian line, he turned against the center as well, heading straight for Darius who was stationed in his own center. Darius panicked as he had done at Issus and fled (Fig. 6, M).

In the meantime, as Alexander drove the routed Persian left wing, the central phalanx found itself unable to follow, especially, as the left Macedonian wing was recoiling before the Persian right. Two units of the phalanx became detached and Persian cavalry broke through Macedonian left center and continued on to loot the Macedonian camp several miles away (Fig. 6, N) rather than wheeling to hit the Macedonian troops in the rear. Darius had no way of ordering his most advantageous units of cavalry to take Alexander's line in the rear as a desire to win instead of looting Macedonian camp. The break-through group attacked the Macedonian baggage train, attempting to rescue Persian prisoners and mowing down the guards. The rear formation of the Macedonian phalanx and light infantry, placed in reserve, saved the situation and drove the enemy from the camp.

At the same time the Macedonian left wing had been overwhelmed by Persians and was on the verge of defeat. Persians fighting against left wing did not know that their king had fled. Alexander received an appeal for help from the left wing commander, abandoned his pursuit of Darius and rode his Companions across the battlefield to save the left wing of his army (Fig. 6, O). He had to fight his way through a crowd of retreating Persians, some of them still in formation, to come to the relief of his left wing. Arrian [2] wrote that "the ensuing struggle was the fiercest of the whole action." By the time Alexander arrived to help, the left wing Macedonian heavy and light cavalry counterattacked, and the whole Persian army was in rout.

Alexander then attempted to catch Darius but it was too late. As at Issus, the Darius example was followed by his entire army. The LG tools will model naturally spreading information that Persian King had abandoned his army and the effect of this information. As in other battles, this will be implemented via communication delays between independent components of LG controlling different parts of the Persian army.

8 Conclusion: What to Expect

We certainly understand that the real test of our hypothesis about the role of LG in human culture will take place only after applying LG software to ancient battles. However, the purpose of this paper is to explain that no mysterious actions or events happened during these historical battles. The decisions of both commanders, the greatest leadership of Alexander and the subpar behavior of Darius and his satraps, have its rational and this rational is well grounded in LG. With proper initial data, including terrain data, appropriate knowledge of the adversarial forces (including appropriate level of misunderstanding), personal traits of the commanders, etc., the

Persian and the Macedonian versions of LG will simulate these battles with the same courses of action as those reported by the historians.

A discovery of the common mathematical foundation of human reasoning about military operations and demonstration of this reasoning taking place throughout human history is mindboggling. Why is it formal, where does it come from? Was it developed unconsciously long before any languages appeared on Earth and recorded originally in the Primary Brain Language described by J. von Neumann [52] and later perfected? Which discoveries in History could be made employing the LG software? Would this kind of experiments make difference for the Evolutionary Psychology, e.g., for the Hunting Hypothesis [2]? Employing LG, are we approaching superintelligence in the domain of warfare? Would an approach analogous to LG, based on the "genetic alphabet" of a hierarchy of building blocks, be beneficial for advancing AI in other domains?

References

1. Anglim, S., Jestice, P., Rice, R., Rusch, S., Serrati, J.: Fighting Techniques of the Ancient World: 3000 BC – 500 AD. Thomas Dunn Books, St Martin's Press, New York (2007)
2. Ardrey, R.: The Hunting Hypothesis: A Personal Conclusion Concerning the Evolutionary Nature of Man, p. 242. Macmillan Pub. Co., Basingstoke (1976)
3. Arrian, Anabasis of Alexander, Books I-IV (Loeb Classical Library No. 236) (1976) (Translated by Brunt, P.)
4. Botvinnik, M.M.: Computers in Chess: Solving Inexact Search Problems. Springer, Heidelberg (1984)
5. Cohen, W.: 25 Ways to Fight Terrorism: PLAYING WAR GAMES. ASEE Prism Magazine, 31 (February 2002)
6. DARPA RAID program, IPTO (Information Processing Technology Office), 2004-08, `http://dtsn.darpa.mil/ipto/programs/raid/raid.asp`
7. Dunnigan, J.: The Complete Wargames Handbook. William Morrow & Co. (1992)
8. Ferrill, A.: The Origins of War: From the Stone Age to Alexander the Great. Rev. ed. Westview Press, Boulder (1997)
9. Frith, C.: Making up the Mind: How the Brain Creates Our Mental World. Blackwell Publishing, Malden (2007)
10. Garcia-Ortiz, A., et al.: Application of Semantic Control to a Class of Pursue-Evader Problems. Computers and Mathematics with Applications 26(5), 97–124 (1993)
11. Hopcroft, J., Motwani, R., Ullman, J.: Introduction to Automata Theory, 3rd edn. Addison Wesley, Reading (2006)
12. Isaacs, R.: Differential Games. Wiley, New York (1965)
13. Knuth, D., Moore, R.: An Analysis of Alpha-Beta Pruning. Artificial Intelligence 6(4), 293–326 (1975)
14. Kott, A.(ed.): Advanced Technology Concepts for Command and Control. Xlibris Corp. (2004)
15. Crick, F.: The genetic code. In: What mad pursuit: a personal view of scientific discovery, ch. 8, pp. 89–101. Basic Books, New York (1988)
16. Kuhn, H.W., Tucker, A.W. (eds.): Contributions to the Theory of Games. Annals of Mathematics Studies, vol. 28, vol. II. Princeton University Press, Princeton (1953)
17. Lee, J., Chen, Y.-L., Yakhnis, V., Stilman, B., Lin, F.: Discrete Event Control and Hostile Counteraction Strategies, DARPA JFACC Final Report, Rockwell Science Center (February 2001)

18. Lirov, Y., Rodin, E.Y., McElhaney, B.G., Wilbur, L.W.: Artificial Intelligence Modeling of Control Systems. Simulation 50(1), 12–24 (1988)
19. Linguistic Geometry Tools: LG-PACKAGE, with Demo DVD, 60 pp., STILMAN Advanced Strategies (2010), This brochure and recorded demonstrations are also available at `http://www.stilman-strategies.com`
20. Linguistic Geometry Workshop, with STILMAN's Comments, REPORT, Dstl, Ministry of Defence, Farnborough, UK, p. 17, (February 25-26, 2003)
21. McQuay, W., Stilman, B., Yakhnis, V.: Distributed Collaborative Decision Support Environments for Predictive Awareness. In: Proc. of the SPIE Conference "Enabling Technology for Simulation Science IX", Orlando, FL, USA (2005)
22. Osborn, M., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge (1994)
23. Rodin, E.: Semantic Control Theory. Applied Mathematical Letters 1(1), 73–78 (1988)
24. Stilman, B.: Formation of the Set of Trajectory Bundles. In: Appendix 1 to the book: On the Cybernetic Goal of Games, Botvinnik, M. M., Soviet Radio, Moscow, pp. 70–77 (1975) (in Russian)
25. Stilman, B.: A Formal Language for Hierarchical Systems Control. Int. J. Languages of Design 1(4), 333–356 (1993)
26. Stilman, B.: A Linguistic Approach to Geometric Reasoning. Int. J. of Computers & Mathematics with Applications 26(7), 29–58 (1993)
27. Stilman, B.: Network Languages for Complex Systems. Int. J. of Computers & Mathematics with Applications 26(8), 51–80 (1993)
28. Stilman, B.: Linguistic Geometry for Control Systems Design. Int. J. of Computers and Their Applications 1(2), 89–110 (1994)
29. Stilman, B.: Translations of Network Languages. Int. J. of Computers & Mathematics with Applications 27(2), 65–98 (1994)
30. Stilman, B.: Managing Search Complexity in Linguistic Geometry. IEEE Transactions on Systems, Man, and Cybernetics 27(6), 978–998 (1997)
31. Stilman, B.: Network Languages for Concurrent Multi-agent Systems. Intl. J. of Computers & Mathematics with Applications 34(1), 103–136 (1997)
32. Stilman, B.: Linguistic Geometry: From Search to Construction, p. 416. Kluwer Acad. Publishers, Dordrecht (2000) (now Springer)
33. Stilman, B.: From Games to Intelligent Systems. In: Proc. of the 2nd ICSC Int. Symp. on Engineering of Intelligent Systems – EIS 2000, University of Paisley, UK, June 27-30, pp. 779–786 (2000)
34. Stilman, B., Dyer, D.: Linguistic Geometry for aerospace combat simulation: serial and concurrent agents. In: Proc. 5th Int. Conf. on Human-Machine Interaction and Artificial Intelligence in Aerospace (HMI-AI-AS 1995), Toulouse, France (1995)
35. Stilman, B., Yakhnis, V.: Solving Adversarial Control Problems with Abstract Board Games and Linguistic Geometry (LG) Strategies. In: Proc. of the 1st Symp.: Advances in Enterprise Control, JFACC Program, DARPA, ISO, San Diego, CA, USA, November 15-16, pp. 11–23 (1999)
36. Stilman, B., Yakhnis, V.: Adapting the Linguistic Geometry – Abstract Board Games Approach to Air Operations. In: Proc. of the 2nd Symp.: Advances in Enterprise Control, JFACC Program, DARPA, Information Systems Office, Minneapolis, MN, USA, July 10-11, pp. 219–234 (2000)
37. Stilman, B., Yakhnis, V.: LG War Gaming for Effects Based Operations, STILMAN Advanced Strategies, Tech. Report to Rockwell and Boeing, p. 47 (July 2001)
38. Stilman, B., Yakhnis, V.: Linguistic Geometry: New Technology for Decision Support. In: Proc. of the SPIE Conference "Enabling Technology for Simulation Science VII", Orlando, FL, April 22-25 (2003)

39. Stilman, B., Yakhnis, V.: LG War Gaming for Effects Based Operations, STILMAN Advanced Strategies, Tech. Report to Rockwell and Boeing, p. 47 (July 2001)
40. Stilman, B., Yakhnis, V.: LG Anti-CM Defense Project in September-December 2001, STILMAN Advanced Strategies, Final Report to Boeing with raw data and charts, p. 35 (January 2002)
41. Stilman, B., Yakhnis, V., Umanskiy, O.: Winning Strategies for Robotic Wars: Defense Applications of Linguistic Geometry. Artificial Life and Robotics 4(3) (2000)
42. Stilman, B., Yakhnis, V., Umanskiy, O.: Knowledge Acquisition and Strategy Generation with LG Wargaming Tools. International Journal of Computational Intelligence and Applications 2(#4), 385–409 (2002)
43. Stilman, B., Yakhnis, V., Umanskiy, O., Hearing, J.: Operational Level Decision Aids with LG-based Tools. In: Proc. of the SPIE Conference "Enabling Technology for Simulation Sci. VI", Orlando, FL, April 1-5 (2002)
44. Stilman, B., Yakhnis, V., McCrabb, M.: LG Wargaming Tool for Effect Based Operations. In: Proc. of the SPIE Conference "Enabling Technology for Simulation Science VI", Orlando, FL, April 1-5 (2002)
45. Stilman, B., Yakhnis, V., Umanskiy, O., Boyd, R.: Adversarial Reasoning and Resource Allocation: The LG Approach. In: Proc. of the SPIE Conference "Enabling Technology for Simulation Science IX", Orlando, FL, USA (2005)
46. Stilman, B., Yakhnis, V., Curry, P., Umanskiy, O.: Deception Discovery and Employment with Linguistic Geometry. In: Proc. of the SPIE Conference "Enabling Technology for Simulation Science IX", Orlando, FL, USA (2005)
47. Stilman, B., Yakhnis, V., Umanskiy, O., Boyd, R.: Linguistic Geometry for Technologies Procurement. In: Proc. of the SPIE Conference "Enabling Technology for Simulation Science IX", Orlando, FL, USA (2005)
48. Stilman, B., Yakhnis, V., Umanskiy, O., Boyd, R.: LG Based Decision Aid for Naval Tactical Action Officer's (TAO) Workstation. In: Proc. of the SPIE Conference "Enabling Technology for Simulation Science IX", Orlando, FL, USA (2005)
49. Stilman, B., Yakhnis, V., Umanskiy, O.: Strategies in Large Scale Problems In: Kott, A., McEneaney, W. (eds.) Adversarial Reasoning: Computational Approaches to Reading the Opponent's Mind, ch. 3.3., pp. 251–285. Chapman & Hall/CRC (2007)
50. Stilman, B., Yakhnis, V., Umanskiy, O.: Linguistic Geometry: Theory and Experiments. In: Proc. of the 3d Int. Workshop on AI in Science and Technology – AISAT 2009, Hobart, Tasmania, Australia, p. 6 (2009)
51. Von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton Univ. Press, Princeton (1944)
52. Von Neumann, J.: The Computer and the Brain. Yale University Press, New Haven (1958)
53. Walker, P.: A Chronology of Game Theory (2001), `http://www.econ.canterbery.ac.nz/hist.htm`
54. Warry, J.: Warfare in the Classical World. Salamander Books, Ltd., London (1998)
55. Weber, R., Stilman, B., Yakhnis, V.: Extension of the LG Hypergame to "Inner Games" Played over the Topology of Competing "Mind Nets". In: Proc. of the SPIE Conference "Enabling Technology for Simulation Science IX, Orlando, FL, USA (2005)
56. Yakhnis, V., Stilman, B.: Foundations of Linguistic Geometry: Complex Systems and Winning Conditions. In: Proceedings of the First World Congress on Intelligent Manufacturing Processes and Systems (IMP&S) (February 1995)
57. Yakhnis, V., Stilman, B.: A Multi-Agent Graph-Game Approach to Theoretical Foundations of Linguistic Geometry. In: Proc. of the Second World Conference on the Fundamentals of Artificial Intelligence, WOCFAI 1995, Paris, France (July 1995)

Elkan's k-Means Algorithm for Graphs

Brijnesh J. Jain and Klaus Obermayer

Berlin Institute of Technology, Germany
{jbj,oby}@cs.tu-berlin.de

Abstract. This paper proposes a fast k-means algorithm for graphs based on Elkan's k-means for vectors. To accelerate the k-means algorithm for graphs without trading computational time against solution quality, we avoid unnecessary graph distance calculations by exploiting the triangle inequality of the underlying distance metric. In experiments we show that the accelerated k-means for graphs is faster than k-means for graphs provided there is a cluster structure in the data.

Keywords: clustering, k-means, graph matching.

1 Introduction

Many real-world objects are represented not as vectors but as attributed graphs, including point patterns, sequences, and trees as special cases. Examples of such objects are protein structures, chemical compounds, high-level descriptions of images, and natural language texts. In comparison to vector spaces, the domain of graphs suffers from sufficient mathematical structure. As a consequence, pairwise clustering algorithms are one of the most widely used methods to partition a given sample of graphs, because they can be applied to patterns from any distance space. The k-medoids algorithm is a well-known alternative that can also be applied to patterns from an arbitrary distance space. The k-medoids algorithm operates like k-means, but replaces the concept of mean by the set median of a cluster [16]. With the emergence of the generalized median [1,4] and sample mean of graphs [8,9], variants of the k-means algorithm have been extended to the domain of graphs [4,8,9].

In an unmodified form, however, pairwise clustering, k-medoids as well as the extended k-means algorithms are slow in practice for large datasets of graphs. The main obstacle is that determining a graph distance is well known to be a graph matching problem of exponential complexity. But even if we resort to graph matching algorithms that approximate graph distances in polynomial time, application of clustering algorithms for large datasets of graphs is still hindered by their prohibitive computational time.

For pairwise clustering, the number of NP-hard graph distance calculations depends quadratically on the number of the input patterns. In the worst-case, when almost all patterns are in one cluster, k-medoids also has quadratic complexity in the number of distance calculations. If the N graph patterns are uniformly distributed in k clusters, k-medoids requires $O(tN^2/k)$ graph distance calculations, where t is the number of iterations required. For k-means, we require kN graph distance calculations at each iteration in order to assign N pattern graphs to their closest centroids. Recomputing the centroids requires additional graph distance calculations. In the best case, when using the incremental arithmetic mean method [10] for approximating a sample mean, N graph distance calculations at each iteration are necessary to recompute the centroids. This gives a total of $t(k+1)N$ graph distance calculations, where t is the number of iterations required. In view of the exponential complexity of the graph matching problem, reducing the number of distance calculations in order to make k-means for graphs applicable is imperative.

In this contribution, we address the important issue to accelerate clustering algorithms. We augment k-means for graphs as proposed by [9,11] with Elkan's idea [3] for accelerating k-means for vectors. For this we assume that the underlying graph distance is a metric. To avoid computationally expensive graph distance calculations, we exploit the triangle inequality by keeping track of upper and lower bounds between input graphs and centroids. Experiments show that k-means for graphs and its accelerated version perform comparable with respect to solution quality, whereas the accelerated version outperforms standard k-means for graphs with respect to computation time.

2 K-Means for Graphs

We briefly review the k-means algorithm for graphs proposed by [9,11].

2.1 Graph Spaces

This section describes the data we want to cluster and introduces a widely used graph metric.

Let $\mathbb{E}$ be a r-dimensional Euclidean vector space. An (*attributed*) *graph* is a triple $X = (V, E, \alpha)$ consisting of a finite nonempty set V of *vertices*, a set $E \subseteq V \times V$ of *edges*, and an *attribute function* $\alpha : V \times V \rightarrow \mathbb{E}$, such that $\alpha(i,j) = \mathbf{0}$ for each pair of distinct vertices i, j with $(i,j) \notin E$.

For simplifying the mathematical treatment, we assume that all graphs are of order n, where n is chosen to be sufficiently large. Graphs of order less than n, say $m < n$, can be extended to order n by including isolated vertices with attribute zero. For practical issues, it is important to note that limiting the maximum order to some arbitrarily large number n and extending smaller graphs to graphs of order n are purely technical assumptions to simplify mathematics. For machine learning problems, these limitations should have no practical impact, because neither the bound n needs to be specified explicitly nor an extension of all graphs to an identical order needs to be performed. When applying the theory, all we actually require is that the graphs are finite.

A graph X is completely specified by its *matrix representation* $\boldsymbol{X} = (\boldsymbol{x}_{ij})$ with elements $\boldsymbol{x}_{ij} = \alpha(i,j)$ for all $1 \leq i, j \leq n$. Let $\mathcal{X} = \mathbb{E}^{n \times n}$ be the Euclidean space of all $(n \times n)$-matrices and let $\mathcal{T}$ denote a subgroup of the set $\mathcal{P}^n$ of all $(n \times n)$-permutation matrices. Two matrices $\boldsymbol{X} \in \mathcal{X}$ and $\boldsymbol{X'} \in \mathcal{X}$ are said to be equivalent, if there is a permutation matrix $P \in \mathcal{T}$ such that $\boldsymbol{P}^{\mathsf{T}}\boldsymbol{X}\boldsymbol{P} = \boldsymbol{X'}$. The quotient set

$$\mathcal{X}_{\mathcal{T}} = \mathcal{X}/\mathcal{T} = \{[\boldsymbol{X}] \,:\, \boldsymbol{X} \in \mathcal{X}\}$$

is a *graph space* of all *abstract graphs* $[\boldsymbol{X}]$ over the *representation space* $\mathcal{X}$ induced by the transformation group $\mathcal{T}$. Note that an abstract graph is an equivalence class of matrices representing the same structure.

Notations: *By concatenating the columns of* $\boldsymbol{X}$*, we obtain a* vector representation $\boldsymbol{x}$ *of* X*. In the remainder of this contribution, we identify* $\mathcal{X}$ *with* $\mathbb{E}^N$ *(*$N = n^2$*) and consider vector- rather than matrix representations of abstract graphs. Thus we write* $\boldsymbol{x}$ *instead of* $\boldsymbol{X}$ *and* $[\boldsymbol{x}]$ *instead of* $[\boldsymbol{X}]$*. We sometimes identify* X *with* $[\boldsymbol{x}]$ *and write* $\boldsymbol{x} \in X$ *instead of* $\boldsymbol{x} \in [\boldsymbol{x}]$*.*

In order to cluster graphs, we equip the graph space $\mathcal{X}_{\mathcal{T}}$ with a metric. Let $\|\cdot\|$ be a Euclidean norm on $\mathcal{X}$. Then the distance function

$$d(X, Y) = \min\{\|\boldsymbol{x} - \boldsymbol{y}\| \,:\, \boldsymbol{x} \in X, \boldsymbol{y} \in Y\},$$

is a metric [11]. A pair $(\boldsymbol{x}, \boldsymbol{y}) \in X \times Y$ of vector representations is called *optimal alignment* if $\|\boldsymbol{x} - \boldsymbol{y}\| = d(X, Y)$. Thus, we have $d(X, Y) \leq \|\boldsymbol{x} - \boldsymbol{y}\|$ for all vector representations $\boldsymbol{x} \in X$ and $\boldsymbol{y} \in Y$, where equality holds if and only if $\boldsymbol{x}$ and $\boldsymbol{y}$ are optimally aligned.

It has been shown that calculating the graph metric $d(X, Y)$ is a NP-hard problem [5], which is referred to as the graph matching problem.

2.2 The k-Means Algorithm for Graphs

Let $\mathcal{X}_{\mathcal{T}}$ be a graph space over some Euclidean space $\mathcal{X}$, where d is the graph metric induced by the Euclidean metric on $\mathcal{X}$. Suppose that $\mathcal{S}_{\mathcal{T}} = \{X_1, \dots, X_N\}$ is a training sample of N graphs drawn from $\mathcal{X}_{\mathcal{T}}$. The k-means clustering algorithm for graphs aims at finding k centroids $\mathcal{Y} = \{Y_1, \dots, Y_k\} \subseteq \mathcal{X}_{\mathcal{T}}$ and a membership matrix $\boldsymbol{M} = (m_{ij})$ such that the cluster objective

$$J\left(\boldsymbol{M}, \mathcal{Y}_{\mathcal{T}} \,|\, \mathcal{S}_{\mathcal{T}}\right) = \frac{1}{N}\sum_{j=1}^{k}\sum_{i=1}^{N} m_{ij}\, d(X_i, Y_j)^2,$$

is minimized with respect to the constraints

$$\begin{aligned} &\sum_{j=1}^{k} m_{ij} = 1 \quad \forall\, i \in \{1, \dots, N\} \\ &m_{ij} \in \{0, 1\} \quad \forall\, i \in \{1, \dots, N\}, \forall\, j \in \{1, \dots, k\}\,. \end{aligned}$$

Necessary conditions for optimality are as follows:

1. *Nearest Neighbor Condition*: Given a fixed set $\mathcal{Y}_{\mathcal{T}}$ of k centroids, the membership matrix $\boldsymbol{M} = (m_{ij})$ satisfies
$$m_{ij} = 1 \;\Leftrightarrow\; j = \arg \min_{1 \leq c \leq k} d\left(X_i, Y_c\right) \qquad \forall\, i \in \{1, \ldots, N\}\,.$$
2. *Centroid Condition*: Given a fixed membership matrix $\boldsymbol{M} = (m_{ij})$, the set $\mathcal{Y}_{\mathcal{T}}$ of k centroids are the structural versions of the sample graph means
$$Y_j = \arg \min_{Y \in \mathcal{X}_{\mathcal{T}}} \sum_{i=1}^{N} m_{ij}\, d(X_j, Y)^2 \qquad \forall\, j \in \{1, \ldots, k\}\,.$$

For a proof we refer to [12]. Note that a sample mean of graphs is invariant under scalings of the sum of squared distances to the sample graphs. The two necessary conditions for optimality form the basis of the k-means procedure to minimize the cluster objective.

It remains to show how to determine a sample mean of graphs. Several different subgradient methods for approximating a sample mean have been suggested [10]. A fast approach to approximate a sample mean of graphs is the *incremental arithmetic mean*. The incremental arithmetic mean method emulates the incremental calculation of the standard sample mean. A sample mean of a sample $X_1, \ldots, X_t$ is approximated according to the rule

$$\begin{aligned} \boldsymbol{y}_1 &= \boldsymbol{x}_1 \\ \boldsymbol{y}_i &= \frac{i-1}{i}\, \boldsymbol{y}_{i-1} + \frac{1}{i}\, \boldsymbol{x}_i \qquad \text{for } 1 < i \leq t \end{aligned}$$

where $(\boldsymbol{x}_i, \boldsymbol{y}_{i-1}) \in X_i \times Y_{i-1}$ are optimal alignments. The graph Y represented by the vector $\boldsymbol{y}_t$ approximates a sample mean of the graphs $X_1, \ldots, X_t$.

In each iteration, the k-means for graphs requires kN distance calculations to assign each pattern graph to a centroid and at least additional $O(N)$ distance calculations for recomputing the centroids using the incremental arithmetic mean subgradient method. This gives a total of at least $O((k+1)N)$ graph distance calculations in each iteration, each of which is NP-hard.

3 Elkan's k-Means for Graphs

This section proposes an accelerated version of k-means of graphs by exploiting the properties of a metric as suggested by Elkan [3].

Frequent evaluations of NP-hard graph distances dominate the computational cost of k-means for graphs. Accelerating k-means therefore aims at reducing the number of graph distance calculations. In [3], Elkan suggested an accelerated formulation of the standard k-means algorithm for vectors exploiting the triangle inequality of the underlying distance metric. Since the graph distance function

d on $\mathcal{X}_\mathcal{T}$ induced by an Euclidean metric is also a metric [11], we can transfer Elkan's idea from Euclidean spaces to graph spaces.

To extend Elkan's k-Means acceleration to graph spaces, we assume that $X \in \mathcal{S}_\mathcal{T}$ is a pattern graph and $Y, Y' \in \mathcal{Y}_\mathcal{T}$ are centroids. By Y_X we denote the centroid closest to pattern graph X. Elkan's acceleration is based on two observations:

1. From the triangle inequality of a metric follows
$$u(X) \leq \frac{1}{2} d(Y_X, Y) \;\Rightarrow\; d(X, Y_X) \leq d(X, Y), \tag{1}$$
where $u(X) \geq d(X, Y_X)$ denotes an upper bound of the distance $d(X, Y_X)$.
2. We have
$$u(X) \leq l(X, Y) \;\Rightarrow\; d(X, Y_X) \leq d(X, Y), \tag{2}$$
where $l(X, Y) \leq d(X, Y)$ denotes a lower bound of the distance $d(X, Y)$.

As an immediate consequence, we safely can avoid to calculate a distance $d(X, Y)$ between a pattern graph X and an arbitrary centroid Y if at least one of the following conditions is satisfied

(C_1) $Y = Y_X$
(C_2) $u(X) \leq \frac{1}{2} d(Y_X, Y)$
(C_3) $u(X) \leq l(X, Y)$

We say, Y is a *candidate centroid* for X if all conditions (C_1)-(C_3) are violated. Conversely, if $Y \neq Y_X$ is not a candidate centroid for X, then either condition (C_2) or condition (C_3) is satisfied. From the inequalities (1) and (2) follows that Y can not be a centroid closest to X. Therefore, it is not necessary to calculate the distance $d(X, Y)$. In the case that Y_X is the onliest candidate centroid for X all distance calculations $d(X, Y)$ with $Y \in \mathcal{Y}_\mathcal{T}$ can be skipped and X must remain assigned to Y_X.

Now suppose that $Y \neq Y_X$ is a candidate centroid for X. Then we apply the technique of "delayed (distance) evaluation". We first test whether the upper bound $u(X)$ is out-of-date, i.e. if $u(X) \gneqq d(X, Y_X)$. If $u(X)$ is out-of-date we improve the upper bound by setting $u(X) = d(X, Y_X)$. Since improving $u(X)$ might eliminate Y as being a candidate centroid for X, we again check conditions (C_2) and (C_3). If both conditions are still violated despite the updated upper bound $u(X)$, we have the following situation

$$u(X) = d\left(X, Y_X\right) > \frac{1}{2} d\left(Y_X, Y\right)$$
$$u(X) = d\left(X, Y_X\right) > l(X, Y).$$

Since the distances on the left and right hand side of the inequality of condition (C_2) are known, we may conclude that the situation for condition (C_2) can not be altered. Therefore, we re-examine condition (C_3) by calculating the distance

Algorithm 1: Elkan's k-Means Algorithm for Structure Spaces

```
01  choose set $\mathcal{Y}_\mathcal{T} = \{Y_1, \dots, Y_k\}$ of initial centroids
02  set $l(X,Y) = 0$ for all $X \in \mathcal{S}_\mathcal{T}$ and for all $Y \in \mathcal{Y}_\mathcal{T}$
03  set $u(X) = \infty$ for all $X \in \mathcal{S}_\mathcal{T}$
04  randomly assign each $X \in \mathcal{S}_\mathcal{T}$ to a centroid $Y_X \in \mathcal{Y}_\mathcal{T}$
05  repeat
06      compute $d(Y, Y')$ for all centroids $Y, Y' \in \mathcal{Y}_\mathcal{T}$
07      for each $X \in \mathcal{S}_\mathcal{T}$ and $Y \in \mathcal{Y}_\mathcal{T}$ do
08          if $Y$ is a candidate centroid for $X$
09              if $u(X)$ is out-of-date
10                  update $u(X) = d(X, Y_X)$
11                  update $l(X, Y_X) = u(X)$
12              if $Y$ is a candidate centroid for $X$
13                  update $l(X,Y) = d(X,Y)$
14                  if $l(X,Y) \leq u(X)$
15                      update $u(X) = l(X,Y)$
16                      replace $Y_X = Y$
17      recompute mean $\check{Y}$ for all $Y \in \mathcal{Y}_\mathcal{T}$
18      compute $\delta(Y) = d\big(Y, \check{Y}\big)$ for all $Y \in \mathcal{Y}_\mathcal{T}$
19      set $u(X) = u(X) + \delta(Y_X)$ for all $X \in \mathcal{X}_\mathcal{T}$
20      set $l(X,Y) = \max\{l(X,Y) - \delta(Y), 0\}$ for all $X \in \mathcal{X}_\mathcal{T}$ and for all $Y \in \mathcal{Y}_\mathcal{T}$
21      replace $Y$ by $\check{Y}$ for all $Y \in \mathcal{Y}_\mathcal{T}$
22  until some termination criterion is satisfied.
```

Remark: Setting the upper and lower bounds in line 19 and 20 implicitly declares them as out-of-date. Updating the bounds with $d(X, Y_X)$ and $d(X,Y)$, resp., implicitly declares them as up-to-date.

$d(X,Y)$ and updating the lower bound $l(X) = d(X,Y)$. If condition (C_3) is still violated, we have

$$u(X) = d(X, Y_X) > d(X,Y) = l(X,y).$$

This implies that X is closer to centroid Y than to Y_X and therefore has to be assigned to centroid Y.

Crucial for avoiding distance calculations are good estimates of the lower and upper bounds $l(X,Y)$ and $u(X)$ in each iteration. For this, we compute the change $\delta(Y)$ of each centroid Y by the distance $\delta(Y) = d(Y, \check{Y})$, where $\check{Y}$ is the recomputed centroid. Based on the triangle inequality, we set the bounds according to the following rules

$$l(X,Y) = \max\{l(X,Y) - \delta(Y), 0\} \tag{3}$$

$$u(X) = u(X) + \delta(Y_X). \tag{4}$$

Table 1. Summary of main characteristics of the data sets

data set	#(graphs)	#(classes)	avg(nodes)	max(nodes)	avg(edges)	max(edges)
letter	750	15	4.7	8	3.1	6
grec	528	22	11.5	24	11.9	29
fingerprint	900	3	8.3	26	14.1	48
molecules	100	2	24.6	40	25.2	44

In addition, $u(X)$ is then declared as out-of-date.[1] Both rules guarantee that $l(X, Y)$ is always a lower bound of $d(X, Y)$ and $u(X)$ is always an upper bound of $d(X, Y_X)$.

Algorithm 1 presents a detailed description of Elkan's k-means algorithm for graphs. During each iteration, $k(k-1)/2$ pairwise distances between all centers must be recomputed (Algorithm 1, line `06`). Recomputing the centroids using incremental arithmetic mean (see Section 2.4) requires additional $O(N)$ distance calculations (Algorithm 1, line `17`). To update the lower and upper bounds, k distances between the current and the new centroids must be calculated (Algorithm 1, line `18`). This gives a minimum of $O\left(N + k^2\right)$ distance calculations at each iteration ignoring the delayed distance evaluations in line `10` and `13` of Algorithm 1. As the centroids converge, we expect that the partition of the training sample becomes more and more stable, which results in a decreasing number of delayed distance evaluations.

4 Experiments

This section reports the results of running k-means and Elkan's k-means on four graph data sets.

Data. We selected four publicly available benchmark data sets described in [15]. Each data set is divided into a training, validation, and a test set. In all four cases, we considered data from the test set only. Table 1 provides a summary of the main characteristics of the data sets. The letter (high) graphs represent distorted letter drawings from the Roman alphabet that consist of straight lines only. The GREC data set [2] consists of graphs representing symbols from architectural and electronic drawings. The fingerprint graphs represent fingerprint images of the NIST-4 database [18]. The molecules data set originally compiled by [13] consists of chemical molecules from two classes (mutagen, non-mutagen).

General Experimental Setup. In all experiments, we applied standard k-means for graphs (std) and Elkan's k-means for graphs (elk) to the aforementioned data sets using the following experimental setup:

To initialize the k-means algorithms, we used a modified version of the "furthest first" heuristic [6]. For each data set $\mathcal{S}$, the first centroid Y_1 is initialized to

[1] In the original formulation of Elkan's algorithm for feature vectors, the upper bounds $u(X)$ are declared as out-of-date regardless of the value $\delta(Y)$.

be a graph closest to the sample mean of $\mathcal{S}$. Subsequent centroids are initialized according to

$$Y_{i+1} = \arg\max_{X \in \mathcal{S}} \min_{Y \in \mathcal{Y}_i} D(X, Y),$$

where $\mathcal{Y}_i$ is the set of the first i centroids. We terminated both k-means algorithms after 3 iterations without improvement of the cluster objective J.

For graph distance calculations and finding optimal alignments, we applied a depth first search algorithm on the letter data set and the graduated assignment [5] on the grec, fingerprint, and molecule data set. The depth first search method guarantees to return optimal solutions and therefore can be applied to small graphs only. Graduated assignment returns approximate solutions.

We used the following measures to assess the performance of an algorithm on a dataset: (1) error (value of the cluster objective J), (2) classification accuracy, (3) silhouette index, and (4) number of graph distance calculations.

The silhouette index [17] is a cluster validation index taking values from $[-1, 1]$. Higher values indicate a more compact and well separated cluster structure. Elkan's k-means incurs a computational overhead to create and update auxiliary data structures and to compute Euclidean distances. This overhead is negligible compared to the time spent on graph distance calculations. Therefore, we report number of graph distance calculations rather than clock times as a performance measure for speed.

Results I: Performance Comparison. We applied standard k-means (std) and Elkan's k-means (elk) to all four data sets in order to assess and compare their performance. The number k of centroids as shown in Table 2 was chosen by compromising a satisfactory classification accuracy against the silhouette index. For each data set 5 runs of each algorithm were performed and the best cluster result selected.

Table 2 summarizes the results. The first observation to be made is that the solution quality of std and elk is comparable with respect to error, classification accuracy, and silhouette index. Deviations are due to the non-uniqueness of the

Table 2. Results of different k-means clusterings on four data sets. Columns labeled with *std* and *elk* give the performance of standard k-means for graphs and Elkan's k-means for graphs, respectively. Rows labeled *matchings* give the number of distance calculations $\left(\times 10^3\right)$, and rows labeled *speedup* show how many times an algorithm is faster than standard k-means for graphs.

	letter		grec		fingerprint		molecules	
	$k = 30$		$k = 33$		$k = 60$		$k = 10$	
measure	std	elk	std	elk	std	elk	std	elk
error	11.6	11.5	32.7	32.2	1.88	1.70	27.6	27.2
accuracy	0.86	0.86	0.84	0.83	0.81	0.82	0.69	0.70
silhouette	0.38	0.39	0.40	0.44	0.32	0.31	0.03	0.04
matchings $\left(\times 10^3\right)$	488	43	198	63	549	52	14	15
speedup	1.0	11.5	1.0	3.1	1.0	10.5	1.0	0.94

sample mean and the approximation errors of the graduated assignment algorithm. The second observation to be made from Table 2 is that elk outperforms std with respect to computation time on the letter, grec, and fingerprint data set. On the molecule data set, std and elk have comparable speed performance. Remarkably, elk requires slightly more distance calculations than std.

Contrasting the silhouette index and the dimensionality of the data to the speedup factor gained by elk, we make the following observation: First, the silhouette index for the letter, grec, and fingerprint data set are roughly comparable and indicate a cluster structure in the data, whereas the silhouette index for the molecule data set indicates almost no compact and homogeneous cluster structure. Second, the dimensionality of the vector representations is largest for molecule graphs, moderate for grec graphs, and relatively low for letter and fingerprint graphs. Thus, the speedup factor of elk and gvr apparently decreases with increasing dimensionality and decreasing cluster structure. This behavior is in line with findings in high-dimensional vector spaces [3]. According to [14], there will be little or no acceleration in high dimensions if there is no underlying structure in the data. This view is also supported by theoretical results from computational geometry [7].

Results II: Speedup vs. Number k of Centroids. In this experiment we investigate how the speedup factor of elk depends on the number k of centroids. For this, we restricted to subsets of the letter and fingerprint data sets. We selected 200 graphs uniformly distributed over the four classes A, E, F, and H. From the fingerprint data set we compiled a subset of 300 graphs uniformly distributed over all three classes. For each chosen number k of centroids 10 runs of each algorithm were conducted and the average of all performance measures was taken. The number k is shown in Table 3.

From the results summarized in Table 3, we see that the speedup factor slowly increases with increasing number k of centroids. The results confirm that std and elk perform comparable with respect to solution quality for varying k. As an aside, all k-means algorithms for graphs exhibit a well-behaved performance in the sense that subgradient methods applied to the nonsmooth cluster objective

Table 3. Results of k-means clusterings on a subset of the letter graphs (A, E, F, H) and the fingerprint graphs. Shown are the average values of the performance measures averaged over 10 runs.

			letter (A, E, F, H)						fingerprint			
measure	$k = 40$		$k = 8$		$k = 12$		$k = 16$		$k = 12$		$k = 16$	
	std	elk	std	elk	std	elk	std	elk	std	elk	std	elk
error	6.9	7.0	4.2	4.2	2.7	2.7	2.4	2.4	41.3	41.5	8.1	6.8
accuracy	0.61	0.60	0.82	0.82	0.94	0.94	0.94	0.94	0.59	0.59	0.64	0.64
silhouette	0.26	0.25	0.30	0.30	0.31	0.30	0.22	0.23	0.20	0.21	0.32	0.33
matchings $(\times 10^2)$	140	65	234	71	416	99	544	108	84	67	480	138
speedup	1.0	2.1	1.0	3.3	1.0	4.7	1.0	6.1	1.0	1.3	1.0	3.5

J indeed minimize J in a reasonable way as shown by the decreasing error for increasing k.

5 Conclusion

We extended Elkan's k-means algorithm from vectors to graphs. Elkan's k-means exploits the triangle inequality to avoid graph distance calculations. Experimental results show that standard and Elkan's k-means for graphs perform equally with respect to solution quality, but Elkan's k-means outperforms standard k-means with respect to speed if there is a cluster structure in the data. The speedup factor of both accelerations increases slightly with the number k of centroids. This contribution is a first step in accelerating clustering algorithms that directly operate in the domain of graphs. Future work aims at accelerating incremental clustering methods.

References

1. Jiang, X., Munger, A., Bunke, H.: On Median Graphs:Properties, Algorithms, and Applications. IEEE Trans. PAMI 23(10), 1144–1151 (2001)
2. Dosch, P., Valveny, E.: Report on the second symbol recognition contest. In: Liu, W., Lladós, J. (eds.) GREC 2005. LNCS, vol. 3926, pp. 381–397. Springer, Heidelberg (2006)
3. Elkan, C.: Using the triangle inequality to accelerate k-means. In: ICML 2003 Conference Proceedings, pp. 147–153 (2003)
4. Ferrer, M.: Theory and algorithms on the median graph. application to graph-based classification and clustering, PhD Thesis, Univ. Autònoma de Barcelona (2007)
5. Gold, S., Rangarajan, A.: Graduated Assignment Algorithm for Graph Matching. IEEE Trans. PAMI 18, 377–388 (1996)
6. Hochbaum, D., Shmoys, D.: A best possible heuristic for the k-center problem. Mathematics of Operations Research 10(2), 180–184 (1985)
7. Indyk, P., Amir, A., Efrat, A., Samet, H.: Efficient algorithms and regular data structures for dilation, location and proximity problems. In: FOCS 1999 Conference Proceedings, pp. 160–170 (1999)
8. Jain, B., Wysotzki, F.: Central Clustering of Attributed Graphs. Machine Learning 56, 169–207 (2004)
9. Jain, B., Obermayer, K.: On the sample mean of graphs. In: IJCNN 2008 Conference Proceedings, pp. 993–1000 (2008)
10. Jain, B., Obermayer, K.: Algorithms for the sample mean of graphs. In: Jiang, X., Petkov, N. (eds.) CAIP 2002. LNCS, vol. 5702, pp. 351–359. Springer, Heidelberg (2009)
11. Jain, B., Obermayer, K.: Structure Spaces. Journal of Machine Learning Research 10 (2009)
12. Jain, B., Obermayer, K.: Graph Quantization. arXiv:1001.0921v[1cs.AI] (2010)
13. Kazius, J., McGuire, R., Bursi, R.: Derivation and validation of toxicophores for mutagenicity prediction. Journal of Medicinal Chemistry 48(1), 312–320 (2005)
14. Moore, A.: The anchors hierarchy: Using the triangle inequality to survive high dimensional data. In: UAI 2000 Conference Proceedings, pp. 397–405 (2000)

15. Riesen, K., Bunke, H.: IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning. In: SSPR 2008 Conference Proceedings, pp. 287–297 (2008)
16. Schenker, A., Last, M., Bunke, H., Kandel, A.: Clustering of web documents using a graph model. In: Web Document Analysis: Challenges and Opportunities, pp. 1–16 (2003)
17. Theodoridis, S., Koutroumbas, K.: Pattern Recognition. Elsevier, Amsterdam (2009)
18. Watson, C., Wilson, C.: NIST Special Database 4, Fingerprint Database, National Institute of Standards and Technology (1992)

A Simple Approach to Incorporate Label Dependency in Multi-label Classification

Everton Alvares Cherman, Jean Metz, and Maria Carolina Monard

Institute of Mathematics and Computer Science (ICMC)
University of São Paulo at São Carlos (USP/São Carlos)
P.O. Box 668, Zip Code 13561-970
São Carlos, SP, Brazil
{echerman,metzz,mcmonard}@icmc.usp.br

Abstract. In multi-label classification, each example can be associated with multiple labels simultaneously. The task of learning from multi-label data can be addressed by methods that transform the multi-label classification problem into several single-label classification problems. The binary relevance approach is one of these methods, where the multi-label learning task is decomposed into several independent binary classification problems, one for each label in the set of labels, and the final labels for each example are determined by aggregating the predictions from all binary classifiers. However, this approach fails to consider any dependency among the labels. In this paper, we consider a simple approach which can be used to explore labels dependency aiming to accurately predict label combinations. An experimental study using decision trees, a kernel method as well as Naïve Bayes as base-learning techniques shows the potential of the proposed approach to improve the multi-label classification performance.

Keywords: machine learning; multilabel classification; binary relevance; label dependency.

1 Introduction

In traditional supervised learning, each example in the training set is associated into a single-label y_j from a set of disjoint labels L, *i.e.* $y_j \in L$. Thus, the single-label classification task deals with classifying examples into a single label. On the other hand, in multi-label learning each example in the training set is associated with multiple labels simultaneously, and the multi-label classification task deals with classifying examples to a set of labels $Y_i \subseteq L$.

Due to the increasing number of new applications where examples are annotated with more than one label, multi-label classification has attracted the attention of the academic community. Multi-label classification is being used in an increasing number of applications such as semantic annotation of video [1] and image [2] , functional genomic [3] and music categorization into emotions [4], to name just a few.

A good survey on multi-label classification is presented in [5]. According to [5], the methods for multi-label classification can be grouped into two main categories: problem transformation and algorithm adaptation. The first group of methods are algorithm independent and they transform the multi-label classification problem into either one or more single-label classification problems. The second group of methods extend specific learning algorithms in order to handle multi-label data directly.

In this work, we shall concentrate on the Binary Relevance (BR) approach which is one of the common problem transformation methods. In this approach, a multi-label classification problem is decomposed into a multiple independent binary classification problem for each one of the single labels y_j which participates in the multi-label problem. In other words, one binary classification problem is created for each y_j, and each example in the training set is labeled as positive if y_j is an element of the correspondent multi-label example, negative otherwise. The final labels for each multi-label example are determined by aggregating the classification results from all binary classifiers. Any supervised learning algorithm can be used as a base algorithm to generate these classifiers.

However, the BR approach presents two problems. For a large number of labels, *i.e.*, when $|L|$ is large, the BR approach may experience drawbacks from the imbalanced data problem. The reason is that, in this case, it is more likely that at least for some binary classifiers, the number of negative examples that indicate that the specific label is not present, could be much larger than the number of positive examples. As a result, these binary classifiers might predict negative for all instances. Nevertheless, it should be noted that in some cases learning algorithms perform well on imbalanced domains. As shown in [6], it is not fair to directly correlate class imbalance to the loss of performance of learning algorithms. In fact, the problem is also related to the degree of overlapping among the classes. Thus, only imbalanced binary classifiers that do not predict the minority class well need to be treated separately in order to improve their performance. This can be done by using methods to deal with the problem of learning in the presence of class imbalance, as the ones described in [7].

The other problem of the BR approach is that it is unable to exploit dependency between the labels in the set of labels. In this paper, we propose a simple approach, named *BR+*, which can be used to incorporate label dependency aiming to accurately predict label combinations. This approach allows the user to specify different ways of considering the labels in order to explore label dependency, where three of them have been already implemented in *BR+*.

In order to validate the proposed approach, we carried out an empirical evaluation on multi-label data sets from different domains. Results show that, for most of the data sets, the proposed *BR+* approach is able to improve the value of several of the multi-label data measures used in the evaluation.

This work is organized as follows: Section 2 briefly describes multi-label classification and the binary relevance approach. Section 3 describes the approach proposed in this paper and Section 4 comments on other solutions closely related to our proposal. An experimental evaluation is presented in Section 5, and Section 6 concludes.

2 Multi-label Classification and the *BR* Approach

Let D be a training set with N examples $E_i = (\mathbf{x}_i, Y_i)$, where $i = 1..N$. Each instance E_i is associated with a feature vector $\mathbf{x} = (x_{i1}, x_{i2}, \ldots, x_{iM})$ and a subset of labels $Y_i \subseteq L$, where $L = \{y_j : j = 1..q\}$ is the set of q possible labels. This representation is shown in Table 1. Considering this scenario, the multi-label classification task is to induce a classifier H that, given an unlabeled instance $E = (\mathbf{x}, ?)$, is capable of accurately predicting its subset of labels Y, *i.e.*, $H(E) \rightarrow Y$.

Table 1. Multi-label data

	$\mathbf{x}$				Y
E_1	x_{11}	x_{12}	$\ldots$	x_{1M}	Y_1
E_2	x_{21}	x_{22}	$\ldots$	x_{2M}	Y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
E_N	x_{N1}	x_{N2}	$\ldots$	x_{NM}	Y_N

As stated earlier, the *BR* approach can be used to solve the multi-label classification problem. It consists of a problem transformation strategy that decomposes a multi-label classification problem into several distinct single-label binary classification problems, one for each label in the set of L labels. The *BR* approach initially transforms the original training dataset into q binary datasets D_{y_j}, $j = 1..q$, where each D_{y_j} contains all examples of the original dataset, but with a single positive or negative label related to the single label y_j according to the true label subset associated with the example, *i.e.*, positive if the label set contains the label y_j and negative otherwise. The other labels $(y_k, k \neq j)$ are not included in D_{y_j}. After the data transformation, a set of q binary classifiers $H_j(E), j = 1..q$ are constructed using the correspondent training dataset D_{y_j}. In other words, the *BR* approach initially constructs a set of q classifiers — Equation 1:

$$H_{BR} = \{C_{y_j}((\mathbf{x}, y_j)) \rightarrow \lambda_j \in \{0,1\} | y_j \in L : j = 1..q\} \quad (1)$$

For the classification of a new multi-label instance, the algorithm outputs the aggregation of the labels positively predicted by all the independent binary classifiers.

A relevant advantage of the *BR* approach is its low computational complexity compared with other multi-label methods. For a constant number of examples *BR* scales linearly with the size of the label set L. Thus, the *BR* approach is quite appropriate for not very large $|L|$, which is the general case.

The performance of multi-label classifiers can be evaluated using different measures. Some of these measures are adaptations from the single-label classification problem, while others were specifically defined to multi-label tasks. A complete discussion on the performance measures for multi-label classification tasks is out of the scope of this paper, and can be found in [5]. In what follows, we briefly describe the measures used in this paper to compare our proposal with the *BR* approach. The measures are *Hamming Loss*, *Accuracy*, *F-Meassure* and *SubsetAccuracy*, defined by Equations 2 to 5 respectively.

$$HammingLoss(H,D) = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i \Delta Z_i|}{|L|} \tag{2}$$

where Δ represents the symmetric difference between two sets, Y_i is the set of true labels and Z_i is the set of predicted labels. When considering the *Hamming Loss* as the performance measure, the smaller the value, the better the algorithm performance is, with a lower bound equal to zero. For the next measures, greater values indicate better performance.

$$Accuracy(H,D) = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{3}$$

$$F(H,D) = \frac{1}{N}\sum_{i=1}^{N}\frac{2|Y_i \cap Z_i|}{|Z_i| + |Y_i|} \tag{4}$$

$$SubsetAccuracy(H,D) = \frac{1}{N}\sum_{i=1}^{N} I(Z_i = Y_i) \tag{5}$$

where $I(\text{true}) = 1$ and $I(\text{false}) = 0$. *SubsetAccuracy* is a very strict evaluation measure as it requires an exact match of the predicted and the true set of labels.

Besides the performance measures, other measures to quantify the cardinality and the density of multi-label data are needed. The *cardinality*, defined by Equation 6, is the average number of labels associated with each example. The *density*, defined by Equation 7, is the normalized cardinality, *i.e.*, the cardinality divided by the total number of single-labels in the set of possible labels L.

$$CR(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}|Y_i| \tag{6}$$

$$DR(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i|}{|L|} \tag{7}$$

3 An Approach to Incorporate Label Dependency

The proposed approach to incorporate label dependency, called *BR+* (*BR plus*), extends the standard binary relevance approach, which completely ignores any label relationships. The main idea is to propagate the labels from training examples to test examples. The label information propagated from different training examples are then used as the basis for determining the class labels of test examples [8].

The training phase of *BR+* consists of two steps. In the first step, similarly to the standard *BR*, $q = |L|$ binary classifiers are induced, one for each label in L — Figure 1(a). Using this classifiers, *BR+* predicts the λ_j labels, $j = 1..q$ on the corresponding test set, as illustrated in Figure 1(b).

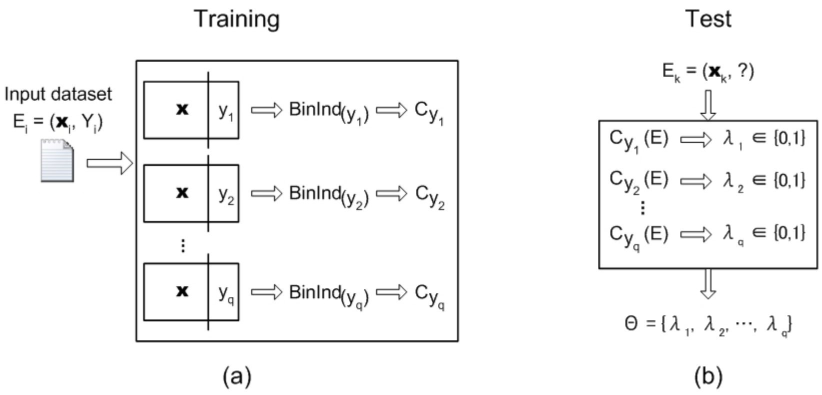

Training
Test
Input dataset
$E_i = (\mathbf{x}_i, Y_i)$
$\mathbf{x}$
y_1
$\Rightarrow$ BinInd$(y_1) \Rightarrow Cy_1$
y_2
$\Rightarrow$ BinInd$(y_2) \Rightarrow Cy_2$
y_q
$\Rightarrow$ BinInd$(y_q) \Rightarrow Cy_q$
$E_k = (\mathbf{x}_k, ?)$
$Cy_1(E) \Rightarrow \lambda_1 \in \{0,1\}$
$Cy_2(E) \Rightarrow \lambda_2 \in \{0,1\}$
$Cy_q(E) \Rightarrow \lambda_q \in \{0,1\}$
$\Theta = \{\lambda_1, \lambda_2, \cdots, \lambda_q\}$
(a)
(b)

Fig. 1. *BR+* first step

In the second step, each training binary dataset is replicated and its feature vector is incremented with $q - 1$ new features, which consist of the labels of the other binary classifiers. In other words, each D_{y_j} binary training dataset is augmented with ω_j binary features where $\omega_j = L - \{y_j\}$. After this transformation *BR+* constructs a set of q new binary classifiers — Equation 8

$$H_{BR+} = \{C_{y_j}((\mathbf{x} \cup \omega_j, y_j)) \rightarrow \lambda_j^{'} \in \{0,1\} | \, \omega_j = L - \{y_j\}, y_j \in L : j = 1..q\} \quad (8)$$

This process is illustrated in Figure 2(a).

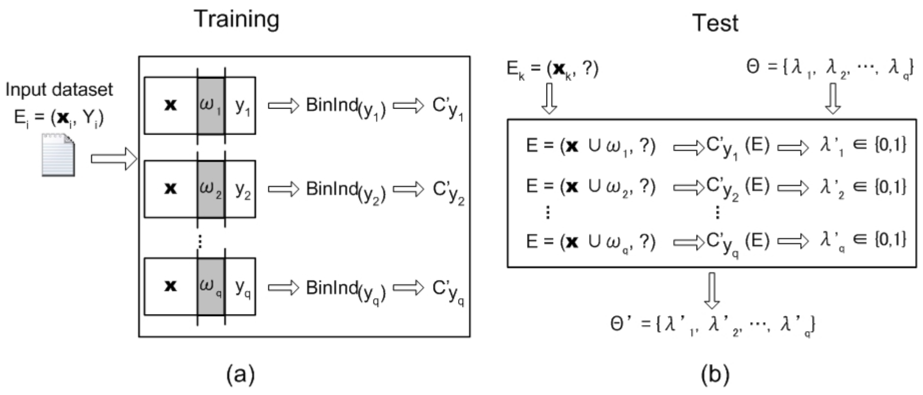

Training
Test
Input dataset
$E_i = (\mathbf{x}_i, Y_i)$
$\mathbf{x}$
ω_1
y_1
$\Rightarrow$ BinInd$(y_1) \Rightarrow C'y_1$
ω_2
y_2
$\Rightarrow$ BinInd$(y_2) \Rightarrow C'y_2$
ω_q
y_q
$\Rightarrow$ BinInd$(y_q) \Rightarrow C'y_q$
$E_k = (\mathbf{x}_k, ?)$
$\Theta = \{\lambda_1, \lambda_2, \cdots, \lambda_q\}$
$E = (\mathbf{x} \cup \omega_1, ?) \Rightarrow C'y_1(E) \Rightarrow \lambda'_1 \in \{0,1\}$
$E = (\mathbf{x} \cup \omega_2, ?) \Rightarrow C'y_2(E) \Rightarrow \lambda'_2 \in \{0,1\}$
$E = (\mathbf{x} \cup \omega_q, ?) \Rightarrow C'y_q(E) \Rightarrow \lambda'_q \in \{0,1\}$
$\Theta' = \{\lambda'_1, \lambda'_2, \cdots, \lambda'_q\}$
(a)
(b)

Fig. 2. *BR+* second step

The classification of a new example could be carried out in a similar way as the standard *BR* approach using this new binary classifiers. However, as each classifier has been trained using a feature space different from the original example's feature vector, the test sets must consider this new feature vector space. Thus, the test sets must be augmented accordingly to the training set that generated the correspondent classifier. However, as in this case the value of the augmented binary features are unknown, the values predicted in the first step of *BR+*, *i.e.* λ_j, are used — Figure 2(b).

At this point, there are different manners to classify a new multi-label instance. Similarly to the standard *BR* approach, *BR+* could output the aggregation of the labels positively predicted by all the binary classifiers constructed in this second step. We shall call this method *NU* (No Update) since each label positively predicted $\lambda_j^{'}$ uses in its corresponding test set the $q-1$ values λ_i, $i \neq j$ as the values of the augmented binary features. The *NU* method is the one illustrated in Figure 2(b).

Nonetheless, if a predefined order to find the labels positively predicted by each classifier is considered, each of these new values $\lambda_j^{'}$ can be used to update the previous λ_j of the correspondent augmented binary feature in the test set. In this work we propose the use of two possible predefined orders taking into account the frequency of single labels in the training set. We shall call this two orderings $F+$, whenever the most frequent labels are considered first, and $F-$, whenever the least frequent labels are considered first. In both cases the predicted values are used to update the correspondent value in the test set.

Regarding *BR+* computational complexity, it can be observed that it simply duplicates *BR* complexity, thus, *BR+* also scales linearly with the size of the label set L.

4 Related Work

Several attempts have been made to explore dependency between the labels in the set of labels using the *BR* approach. In [9], a new method that uses the concept of chains in the *BR* framework is proposed. The method scales up to large datasets and makes an effective use of the label correlation information. A chain of classifiers is formed and each classifier is responsible for learning and predicting the binary association of a specific label, given the feature space augmented by all prior *BR* predictions in the chain, *i.e.*, the feature space of each link in the chain is extended with the $0-1$ (present or not) label associations of all previous links. Regarding the chain order, several possible heuristics can be used, although the effect of different orders on the prediction performance of the algorithm was not studied in depth. The authors solved this issue by using an ensemble framework with a different random chain ordering per iteration. The predictions are summed by label so that each label receives a number of votes. A threshold is used to select the most popular labels which form the final predicted multi-label set.

In [10], a stacked *BR* classification output indicating relationships between classes along with the full original feature space into a separate meta classifier is proposed, thereby creating a two-stage classification process. The method proposed in [10] is similar to the one we are proposing in this paper. However, in [10], the authors use SVMs and propose a novel kernel for their method. We, instead, do not restrict our method for a specific underlying algorithm. Moreover, they use all the outputs of the *BR* classifier to train the meta classifier in the second step. This can be a potential problem, since these new features are actually an estimate of the true label, made in the first *BR* classifier stage. Using this estimated value can lead the solution to an overfitted model. Our proposal, on the other hand, does not use the estimate for the label y_j, *i.e.*, λ_j, when training the individual final binary classifier in the *BR* approach for that label, but only keeps the estimate for the other labels, consequently, maintaining the label correlation information in the set. Furthermore, in our work we include an empirical evaluation of different ways of considering the labels when building individual models in the *BR* approach to effectively use the interrelationships between labels.

5 Experiments and Results

The main objective of our study is to compare the standard *BR* approach with *BR+*, considering the three different methods already implemented, the no update method *NU*, and the two update methods $F+$ and $F-$. The experiment were repeated using three different base single-label classifiers.

5.1 Datasets

To make this comparison, we selected five multi-label datasets from different domains. The characteristics of the datasets, which can be found in the Mulan website[1], are shown in Table 2.

Table 2. Description of the datasets used in the experiments

Dataset	**domain**	**#examples**	**#features**	$\lvert L\rvert$	**CR-cardinality**	**DR-density**	**#distinct**
emotions	music	593	72	6	1.869	0.311	27
medical	text	978	1449	45	1.245	0.028	94
scene	image	2407	294	6	1.074	0.179	15
yeast	biology	2417	103	14	4.237	0.303	198
corel5k	images	5000	499	374	3.522	0.009	3175

5.2 Experimental Setup

BR+ has been implemented as an extension method of Mulan[2], a package of Java classes for multi-label classification based on Weka[3]. The experiments were carried out using three different base single-label classifiers from Weka. J48, a

[1] http://mulan.sourceforge.net/datasets.html
[2] http://mulan.sourceforge.net
[3] http://www.cs.waikato.ac.nz/ml/weka/

decision tree learning algorithm, Naïve Bayes and the support vector machine SMO learning algorithm.

All the reported results were obtained using 10-fold cross validation with paired folds, *i.e.*, the same training and testing partitions were used to obtain the average of the four measures considered to compare the standard *BR* and *BR+*.

5.3 Results

Tables 3 to 5 show, respectively, the average of the four measures values, for each base-classifier used in the experiments. For the sake of visualisation, the best measure values are shown in bold. In all the tables, the numbers in brackets correspond to the standard deviation. Arrows show significance according to Bonferroni-Dunn multiple comparison with *BR* as a control test [11].

In order to analyze whether there is a difference among the methods, for each one of the four measures and the three single-label base-classifier, we ran the Friedman test with the null-hypotheses that the performance of all methods are comparable considering all results. Only three of the 12 cases did not reject the null-hypothesis: Accuracy/SMO, F-Measure/J48 and Hamming Loss/Naïve Bayes — Null-hypothesis not rejected in row Stat. Dif. of Tables 3, 4 and 5 respectively.

Table 3. Average measure values using SMO as single-label base-classifier. ↗ and ↘ indicates statistically significant improvement or degradation.

	Accuracy				F-Measure			
Dataset	*BR*	*BR+(S.A.)*	*BR+(F+)*	*BR+(F−)*	*BR*	*BR+(NU)*	*BR+(F+)*	*BR+(F−)*
Corel5k	0.08(0.01)	**0.15(0.00)**	**0.15(0.00)**	0.12(0.00)	0.11(0.01)	**0.23(0.00)**	0.22(0.00)	0.17(0.00)
Emotions	0.50(0.04)	**0.53(0.07)**	0.52(0.08)	0.50(0.08)	0.58(0.04)	**0.63(0.07)**	0.61(0.09)	0.59(0.08)
Medical	0.74 (0.04)	0.77 (0.04)	0.77(0.04)	**0.78(0.03)**	0.77(0.03)	**0.80(0.03)**	0.79(0.04)	**0.80(0.03)**
Scene	0.58(0.03)	0.65(0.03)	0.65(0.04)	**0.68(0.03)**	0.61(0.03)	0.69(0.03)	0.68(0.04)	**0.70(0.03)**
Yeast	0.49(0.03)	0.48(0.03)	0.50(0.03)	**0.51(0.03)**	0.60(0.02)	0.60(0.02)	**0.61(0.02)**	**0.61(0.02)**
Stat. Dif.	Null-hypothesis not rejected				Control	↗		

	Hamming Loss				Subset Accuracy			
Dataset	*BR*	*BR+(NU)*	*BR+(F+)*	*BR+(F−)*	*BR*	*BR+(NU)*	*BR+(F+)*	*BR+(F−)*
Corel5k	**0.01(0.00)**	0.02(0.00)	**0.01(0.00)**	**0.01(0.00)**	0.01(0.00)	0.01(0.00)	**0.02(0.00)**	0.01 (0.00)
Emotions	**0.20(0.02)**	0.24(0.03)	0.23(0.04)	0.24(0.04)	**0.27(0.06)**	0.22(0.06)	0.24(0.08)	0.25(0.08)
Medical	**0.01(0.00)**	**0.01(0.00)**	**0.01(0.00)**	**0.01(0.00)**	0.66(0.06)	0.69(0.05)	0.69(0.05)	**0.71(0.05)**
Scene	**0.11(0.01)**	0.13(0.01)	0.12(0.01)	**0.11(0.01)**	0.51(0.03)	0.54(0.03)	0.59(0.04)	**0.61(0.03)**
Yeast	**0.20(0.01)**	0.22(0.01)	0.21(0.01)	0.21(0.01)	0.13(0.03)	0.14(0.04)	0.17(0.04)	**0.19(0.04)**
Stat. Dif.	Control	↘			Control			↗

When the null-hypothesis is rejected by the Friedman test, at 95% of confidence level, we can proceed with a post-hoc test to detect which differences among the methods are significant. We ran the Bonferroni-Dunn multiple comparison with a control test, using *BR* as a control. Therefore, the Bonferroni-Dunn test points out whether there is a significant difference among *BR* and the three methods of *BR+* involved in the experimental evaluation.

It has been observed that sometimes the Friedman test reports a significant difference but the post-hoc test fails to detect it. This is due to the lower power of

Table 4. Average measure values using J48 as single-label base-classifier. $\nearrow$ and $\searrow$ indicates statistically significant improvement or degradation.

	Accuracy				F-Measure			
Dataset	*BR*	*BR+(NU)*	*BR+(F+)*	*BR+(F−)*	*BR*	*BR+(NU)*	*BR+(F+)*	*BR+(F−)*
corel5k	0.06(0.00)	**0.09(0.01)**	**0.09(0.01)**	**0.09(0.01)**	0.09(0.01)	**0.13(0.01)**	**0.13(0.01)**	**0.13(0.01)**
Emotions	0.44(0.07)	0.44(0.06)	**0.48(0.07)**	0.47(0.06)	0.53(0.08)	0.53(0.07)	**0.56(0.08)**	0.55(0.06)
Medical	0.76(0.03)	0.78(0.04)	0.79(0.03)	**0.80(0.04)**	0.79(0.04)	0.81(0.04)	0.81(0.03)	**0.82(0.04)**
Scene	0.52(0.03)	0.53(0.04)	**0.57(0.05)**	0.55(0.03)	0.55(0.03)	0.57(0.04)	**0.59(0.05)**	0.57(0.03)
Yeast	0.42(0.02)	0.41(0.02)	**0.45(0.03)**	0.42(0.03)	0.55(0.02)	0.54(0.02)	**0.56(0.03)**	0.53(0.03)
Stat. Dif.	Control		$\nearrow$		Null-hypothesis not rejected			

	Hamming Loss				Subset Accuracy			
Dataset	*BR*	*BR+(NU)*	*BR+(F+)*	*BR+(F−)*	*BR*	*BR+(NU)*	*BR+(F+)*	*BR+(F−)*
corel5k	**0.01(0.00)**	**0.01(0.00)**	**0.01(0.00)**	**0.01(0.00)**	0.00(0.00)	**0.01(0.00)**	**0.01(0.00)**	**0.01(0.00)**
Emotions	**0.24(0.04)**	0.27(0.03)	0.25(0.04)	0.25(0.04)	0.19(0.05)	0.18(0.09)	**0.25(0.08)**	**0.25(0.08)**
Medical	**0.01(0.00)**	**0.01(0.00)**	**0.01(0.00)**	**0.01(0.00)**	0.67(0.04)	0.70(0.04)	0.70(0.03)	**0.71(0.04)**
Scene	**0.14(0.01)**	0.19(0.02)	0.15(0.02)	0.15(0.01)	0.43(0.04)	0.43(0.04)	**0.52(0.05)**	0.51(0.04)
Yeast	**0.25(0.01)**	0.28(0.01)	0.26(0.01)	0.29(0.01)	0.05(0.02)	0.06(0.02)	**0.12(0.03)**	0.11(0.03)
Stat. Dif.	Control				Control		$\nearrow$	$\nearrow$

Table 5. Average measure values using Naïve Bayes as single-label base-classifier. $\nearrow$ and $\searrow$ indicates statistically significant improvement or degradation.

	Accuracy				F-Measure			
Dataset	*BR*	*BR+(NU)*	*BR+(F+)*	*BR+(F−)*	*BR*	*BR+(NU)*	*BR+(F+)*	*BR+(F−)*
Corel5k	0.15(0.00)	0.16(0.01)	**0.17(0.01)**	**0.17(0.01)**	0.22(0.01)	0.23(0.01)	**0.24(0.01)**	**0.24(0.01)**
Emotions	0.53(0.04)	0.56(0.06)	0.56(0.07)	**0.58(0.06)**	0.63(0.03)	0.65(0.06)	0.64(0.07)	**0.66(0.06)**
Medical	0.27(0.04)	**0.42(0.06)**	**0.42(0.06)**	**0.42(0.06)**	0.30(0.04)	**0.46(0.06)**	**0.46(0.06)**	**0.46(0.06)**
Scene	0.45(0.03)	**0.59(0.02)**	**0.59(0.02)**	**0.59(0.02)**	0.56(0.03)	**0.69(0.02)**	**0.69(0.02)**	0.68(0.02)
Yeast	0.43(0.03)	0.43(0.03)	**0.44(0.03)**	**0.44(0.03)**	**0.55(0.02)**	**0.55(0.03)**	**0.55(0.03)**	**0.55(0.03)**
Stat. Dif.	Control		$\nearrow$	$\nearrow$	Control			

	Hamming Loss				Subset Accuracy			
Dataset	*BR*	*BR+(NU)*	*BR+(F+)*	*BR+(F−)*	*BR*	*BR+(NU)*	*BR+(F+)*	*BR+(F−)*
corel5k	**0.01(0.00)**	**0.01(0.00)**	**0.01(0.00)**	**0.01(0.00)**	0.00(0.00)	**0.01(0.00)**	**0.01(0.00)**	**0.01(0.00)**
Emotions	0.25(0.04)	0.21(0.04)	0.22(0.04)	**0.20(0.04)**	0.21(0.07)	0.30(0.07)	0.30(0.08)	**0.33(0.08)**
Medical	**0.02(0.00)**	0.03(0.00)	0.03(0.00)	0.03(0.00)	0.19(0.04)	**0.33(0.07)**	**0.33(0.07)**	**0.33(0.07)**
Scene	0.24(0.01)	**0.16(0.01)**	**0.16(0.01)**	**0.16(0.01)**	0.18(0.04)	**0.32(0.04)**	**0.32(0.04)**	**0.32(0.04)**
Yeast	0.29(0.02)	**0.28(0.02)**	**0.28(0.02)**	**0.28(0.02)**	0.10(0.02)	0.11(0.03)	**0.12(0.03)**	**0.12(0.03)**
Stat. Dif.	Null-hypothesis not rejected				Control		$\nearrow$	$\nearrow$

the post-hoc test. In this case, the only conclusion that can be drawn is that some algorithms do differ significantly [11]. In our experiments this has occurred in the following two cases: Hamming Loss/J48 and F-Measure/Naïve Bayes, where no $\nearrow$ or $\searrow$ are shown. However, it can be observed that in these two cases that Hamming Loss/J48 is better for *BR* while F-Measure/Naïve Bayes is better for *BR+*. In general, Hamming Loss does not improve for *BR+*.

However, the other measure values are better for *BR+* most of the time, and in some cases they are significantly better. To sum up, Table 6 shows the number of times, maximum 15, that Accuracy, F-Measure and Subset Accuracy are better for *BR+* considering the results in all datasets for each single-label classifier.

Comparing now the results obtained using the three different base single-label classifiers in each dataset, it is possible to observe that they have a strong influence. In all cases, very poor results were obtained with dataset Corel5k. In this dataset $|L| = 374$ — Table 2. Furthermore, the majority of the 374 binary classifiers constructed in the first step of *BR+* have zero precision, which

Table 6. Number of wins, maximum 15, of *BR+* over *BR*

	Accuracy	F-Measure	Subset Accuracy
SMO	13	14	10
J48	12	12	13
Naïve Bayes	14	12	15

justifies the poor results obtained using this dataset. Considering Accuracy and F-Measure, Naïve Bayes is a clear winer for datasets Corel5K (although this dataset shows very poor results as already mentioned) and Emotion; J48 for dataset Medical, and SMO por datasets Scene and Yeast. Not considering dataset Corel5k, which shows very bad results for Subset Accuracy, the best results for Subset Accuracy were also obtained by the same datasets and algorithm.

Regarding the three different manners to classify a new multi-label instance, *i.e.*, NU, $F+$ and $F-$, it is not possible to identify a clear winner.

6 Conclusions

This paper presented a simple approach for multi-label classification, which is based on the Binary Relevance approach. Its main objective consists of exploiting label dependency aiming to accurately predict label combinations. Using several multi-label datasets from different domains and several evaluation measures, we carried out an empirical evaluation to compare both approaches and to verify if the proposed approach is able to predict label combinations more accurately. Three different manners to classify a new multi-label instance were used. However, it was not possible to identify a clear winner. Initial results show the potential of the proposed approach to improve the multi-label classification performance. Nevertheless, more extensive experiments should be done in the future in order to validate these results. As future work we plan to investigate other predefined orders to find the labels positively predicted by each binary classifier.

Acknowledgements. This research was supported by the Brazilian Research Councils FAPESP and CNPq.

References

1. Qi, G.J., Hua, X.S., Rui, Y., Tang, J., Mei, T., Zhang, H.J.: Correlative multi-label video annotation. In: MULTIMEDIA 2007: Proceedings of the 15th International Conference on Multimedia, pp. 17–26. ACM, New York (2007)
2. Yang, S., Kim, S.K., Ro, Y.M.: Semantic home photo categorization. IEEE Transactions on Circuits and Systems for Video Technology 17, 324–335 (2007)
3. Blockeel, H., Schietgat, L., Struyf, J., Džeroski, S., Clare, A.: Decision trees for hierarchical multilabel classification: A case study in functional genomics. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 18–29. Springer, Heidelberg (2006)

4. Trohidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.: Multilabel classification of music into emotions. In: ISMIR 2008: 9th International Conference on Music Information Retrieval, pp. 325–330 (2008)
5. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining multi-label data. In: Data Mining and Knowledge Discovery Handbook, pp. 1–19 (2009)
6. Prati, R.C., Batista, G.E., Monard, M.C.: Class imbalance versus class overlaping: an analysis of a learning system behaviour. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H. (eds.) MICAI 2004. LNCS (LNAI), vol. 2972, pp. 312–321. Springer, Heidelberg (2004)
7. Batista, G.E., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explorations 6, 20–29 (2004)
8. Kang, F., Jin, R., Sukthankar, R.: Correlated label propagation with application to multi-label learning. In: CVPR, vol. (2), pp. 1719–1726. IEEE Computer Society, Los Alamitos (2006)
9. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009. LNCS, vol. 5782, pp. 254–269. Springer, Heidelberg (2009)
10. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 22–30. Springer, Heidelberg (2004)
11. Demšar, J.: Statistical comparison of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)

Methods and Algorithms of Information Generalization in Noisy Databases

Vadim Vagin and Marina Fomina

Moscow Power Engineering Institute (Technical University),
Krasnokazarmennaya 14, 111250 Moscow, Russia
vagin@appmat.ru, m_fomina2000@mail.ru

Abstract. The problem of information generalization with account for the necessity of processing the information stored in real data arrays which may contain noise is considered. Noise models are presented, and a noise effect on the operation of generalization algorithms using the methods of building decision trees and forming production rules is developed. The results of program modeling are brought about.

Keywords: knowledge acquisition, information generalization, data mining, knowledge discovery, decision tree, noise model, learning sample.

1 Introduction

Knowledge discovery in databases (DB) is important for many technical, social, and economic problems. Up-to-date DBs contain such a huge quantity of information that it is practically impossible to analyze this information manually to acquire valuable knowledge for decisions making. The systems for automatic knowledge discovery in DB are capable of analyzing "raw" data and presenting the extracted information faster and with more success than an analyst could find it himself. At first we consider setting up the generalization problem and the methods of its decision. Then the noise models and prediction of unknown values in accordance with the nearest neighbour method in learning samples are viewed. And finally modeling the algorithm of decision tree with the combination of forming production rules in the presence of noise and results of program simulation are given.

2 Setting Up the Generalization Problem

Knowledge discovery in DB is closely connected with the solution of the inductive concept formation problem or the generalization problem.

Let us give the formulation of feature-based concept generalization [1].

Let $O = \{o_1, o_2, ..., o_n\}$ be a set of objects that can be represented in an *intelligent decision support system* (IDSS). Each object is characterized by *r* attributes: $A_1, A_2, \ldots, A_r$. Denote by $Dom(A_1), Dom(A_2), \ldots, Dom(A_r)$ the sets of admissible values of features where $Dom(A_k)=\{x_1, x_2, \ldots x_{q_k}\}$, $1 \le k \le r$, q_k is the number of different values of the feature A_k. Thus, each object $o_i \in O$, $1 \le i \le n$ is represented as a set of feature values, i.e., $o_i = \{x_{i1}, x_{i2}, \ldots, x_{ir}\}$, where $x_{ik} \in Dom(A_k)$, $1 \le k \le q_k$. Such a description of an object is called *a feature description*. Quantitative, qualitative, or scaled features can be used as object features [1].

Let O be the set of all objects represented in a certain IDSS; let V be the set of *positive objects* related to some concept and let W be the set of *negative objects*. We will consider the case where $O = V \cup W$, $V \cap W = \varnothing$, $W = \bigcup_i W_i$, and $W_i \cap W_j = \varnothing \ (i \neq j)$. Let K be a non-empty set of objects such that $K = K^+ \cup K^-$, where $K^+ \subset V$ and $K^- \subset W$. We call K a *learning sample*. Based on the learning sample, it is necessary to build a rule separating positive and negative objects of a learning sample.

Thus, the concept is formed if one manages to build a decision rule which, for any example from a learning sample, indicates whether this example belongs to the concept or not. The algorithms that we study form a decision in the form of rules of the type "IF condition, THEN the desired concept." The condition is represented in the form of a logical function in which the boolean variables reflecting the feature values are connected by logical connectives. Further, instead of the notion "feature" we will use the notion "attribute". The decision rule is correct if, in further operation, it successfully recognizes the objects which originally did not belong to the learning sample.

The presence of noise in data changes the above setting up of the generalization problem both at the stage of building decision rules and at the stage of the object classification. First of all, the original learning sample K is replaced by the sample K' in which distorted values or missing values of features occur with a certain probability. We consider the solution of the concept generalization problem using the methods of decision trees [2, 3] with the combination of forming production rules [4, 5].

3 Methods for Knowledge Discovery

A number of methods suitable for knowledge discovery is known. They can be divided into the following groups.

Statistical methods. The statistical approach is usually based on probabilistic models. In general, it is assumed that such methods will be used by specialists in statistics; therefore, human interference is required in generating the hypotheses-candidates and models.

Case-based inference. A case is a situation that arose earlier and was solved. In arising a new situation, case-based inference checks up a multitude of situations stored in DB and finds a similar one.

Neural networks. Neural networks belong to the class of systems modeling processes of data treatment like in a human brain. They consist of a great number of artificial neurons and their important peculiarity is the possibility of learning.

Decision trees. The decision tree is a tree in which each nonfinite node accomplishes checking of some condition, and in case a node is finite, it gives out a decision for the element being considered. In order to perform the classification of the given example, it is necessary to start with the root node. Then, we go along the decision tree from the root to the leaves until the finite node (or a leaf) is reached. In each nonfinite node one of the conditions is verified. Depending on the result of verification, the corresponding branch is chosen for further movement along the tree. The solution is obtained if we reach a finite node.

Inductive rules. The rules establish a statistical correlation between the appearance of certain features in a particular example or between certain examples in a learning set.

Bayesian belief networks. A Bayesian belief network is an oriented acyclic graph the nodes of which are feature values, and the weights of arcs are probabilistic dependencies between attributes. Each node is correlated to conditional probability distributions describing the relations between a node and its ancestors.

Genetic algorithms/ Evolutionary modeling. Genetic algorithms and evolution are algorithmically optimized strategies based on the principles of natural evolution. Genetic algorithms and evolutionary modeling use generalization methods to formulate hypotheses about the dependencies between variables in the form of associative rules or by some other formalism.

Rough sets. A rough set is defined by the assignment of upper and lower boundaries of a certain set called the approximations of this set. Each element of the lower approximation is certainly an element of the set. Each element that does not belong to the upper approximation is certainly not an element of the set. The difference in the upper and lower approximations of a rough set forms the so-called boundary region. The element of the boundary region is probably (but not certainly) an element of the set. Similarly to fuzzy sets, rough sets are mathematical conception for work with fuzziness in data.

4 Noise Models

Assume that examples in a learning sample contain noise, i.e., attribute values may be missing or distorted. Noise arises due to following causes: incorrect measurement of the input parameters; wrong description of parameter values by an expert; the use of damaged measurement devices; and data lost in transmitting and storing the information [6].

We study three noise models:

1. Noise is connected with the absence of attribute values (an attribute value did not manage to measure). For each attribute A, a domain of admissible values includes the value "not known." Such a value corresponding to the situation, when the true value of an attribute has been lost, is denoted by N (Not known). Thus, individual examples of a learning sample K' contain a certain quantity of attributes with the values "not known".

2. Noise is connected with the distortion of certain attribute values in a learning sample. The true value is replaced by one admissible, but wrong, what leads to changing the attribute value distribution. In Tables 3, 4, this noise type is called “permutation”.

3. Noise is connected with the presence of mixing any two different attribute values in two rows of a learning sample. This procedure is referred to one of controlled scrambling [6]. Under such a noise, a value distribution for a "noisy" attribute is retained but some values will be still erroneous. In Tables 3, 4, this noise type is called "scrambling".

Further, we consider the work of the generalization algorithm in the presence of noise in original data. Our purpose is to assess the classification accuracy of examples in a control sample by increasing a noise level in this sample.

5 Methods of Restoring Absent Values in a Learning Sample

Let a sample with noise, K', be given; moreover, let the attributes taking both discrete and continuous values be subject to distortions. Consider the problem of using the objects of a learning sample K' in building a decision tree T and in conducting the examination using this tree.

Let $o_i \in K'$ be an object of the sample; $o_i = <x_{i1}, \ldots , x_{ir}>$. Among all the values of its attributes, there are attributes with the value N. These attributes may be both discrete and continuous.

Building a decision tree while having examples with absent values leads to multivariant decisions. Therefore, we try to find the possibility for restoring these absent values. One of the simplest approaches is a replacement of an unknown attribute value on the average (mean or the mode, *MORM*). Another possible approach is the nearest neighbour method which was proposed for the classification of the unknown object X on the basis of consideration of several objects with the known classification nearest to it. The decision on the assignment of the object X to one or another class is made by information analysis on whether these nearest neighbours belong to one or another class. We can use even a simple count of votes to do this. The given method is implemented in the algorithm of restoring that was considered in detail in [7, 8]. Since the solution in this method explicitly depends on the object quantity, we shall call this method as the search method of k nearest neighbours (*kNN).*

6 Generalization Algorithms

Research of noise effect on the operation of generalization algorithms has been performed on the basis of comparative analyses of two known algorithms C 4.5 and CN2 [2-5]. Given algorithms are learning algorithms "with a teacher": initially, the classifying rules are formed in a learning sample, and then an examination is produced in a test sample, results of which allow to assess the efficiency of the given generalization.

The algorithm C4.5 as its predecessor ID3 suggested by J.R.Quinlan [2, 3] refers to an algorithm class building the classifying rules in the form of decision trees. However, C4.5 works better than ID3 and has a number of advantages:

- numerical (continuous) attributes are introduced;
- nominal (discrete) values of a single attribute may be grouped to perform more difficult checking;

- subsequent shortening after inductive tree building based on using a test set allows to increase a classification accuracy.

The algorithm CN2 [4, 5] is one which forms a set of production rules of the kind ”IF <condition> THEN <class>” (one rule for each class) as a classifier. In the given case a condition represents a test conjunction for attribute values. The empty condition in a rule means that a rule covers all examples of a set. This algorithm looks through the whole object set and finds a rule that is the best among possible ones adding it at the end of a set of formed production rules. However CN2 does not exclude from viewing the rules covering one or several negative examples. After the best rule was found, the algorithm deletes covered examples from a learning sample and repeats a searching procedure of the best rule for rest objects. It halts the work when it is impossible to select a best rule for the current object set or the examples in a learning sample were finished.

In order to avoid forming the rules covering few examples, the algorithm CN2 uses so called checking a condition on the importance which confirms, that a distribution of classes among examples covered by a condition is not random and close to a class distribution among all examples of a learning sample.

$$\text{Significance} = 2\sum_{i=1}^{n} p_i \log({}^{p_i}/_{q_i})$$

where p_i is a probability of appearing a class value among examples satisfying to a condition, and q_i is a probability of appearing a class value among all examples of a learning sample. Under using such a test, the majority of conditions covering few examples are excluded from a consideration because their classification accuracy is perceived by the algorithm as randomness.

7 Modeling the Algorithms of Forming Generalized Notions in the Presence of Noise

The above mentioned algorithms have been used to research the effect of a noise on forming generalized rules and on classification accuracy of test examples. To restore unknown values the methods of nearest neighbours (*kNN*) and choice of average (*MORM*) are used. We propose the algorithm "**I**nduction of **D**ecision **T**ree with restoring **U**nknown **V**alues" (IDTUV2). IDTUV2 includes the procedure of restoring unknown values in the presence of examples containing a noise of the type “*absent values*”. When absent values of attributes are restored, one of algorithms of notion generalization is used.

Below, we present the pseudocode of the IDTUV2 algorithm.

```
Algorithm IDUTV2
 Given: $K = K^+ \cup K^-$
   Obtain: decision rules: decision tree $T$, production rules $R$.
 Beginning
     Obtaining $K = K^+ \cup K^-$
       Select noise model, noise level
```

```
            If noise model = absent values
            Then select Method of restoring absent values;
            Perform: introduce noise in K
            Select generalization algorithm C4.5 or CN2
            If Select C4.5  Then obtain decision tree T
            Else Select CN2 and obtain production rules R
            End if
            Output decision tree T or  production rules R
        End
    End of IDTUV2.
```

8 Modeling the Algorithm of Building the Decision Tree in the Presence of Noise

To develop the generalization system, the instrumental environment MS Visual C# Express Edition has been used. The given environment is a shortened version MS Visual Studio. As DBMS SQLite, realizing the standard SQL92 has been used.

The program IDTUV2 performs the following main functions:

- builds the classification model (a decision tree, or production rules) on the basis of the learning sample;
- forms production rules corresponding to the constructed tree;
- recognizes (classifies) objects using a classification model.

We present experiment results carried out on the following four data groups from the known collection of the test data sets of California University of Informatics and Computer Engineering "UCI Machine Learning Repository" [9]:

1. Data of Monk's problems;
2. Medical data: diagnostics of heart diseases (Heart);
3. Repository of data of the StatLog project:
   - diagnostics of diabetes diseases (Diabetes);
   - australian credit (Australian);
4. Other data sets (from the field of biology and juridical-investigation practice).

8.1 Classification Results of Examples with Noise

To define an attribute, which values are subject to distortions, the following approach was used: we chose the most informative attribute of a table. Obviously, changes in values of such attribute can affect the algorithm results most essentially. When we use the algorithm C4.5 to build a decision tree the most informative attribute is located at the root of a decision tree. When we form a production rule set by the algorithm CN2, we choose an attribute used in the first production. When we introduce noise of the type “scrambling” and “permutation”, it is necessary to have a “noisy” attribute being discrete. Therefore, if a most informative attribute is continuous, then preliminarily it needs to perform discretization.

8.2 Assessment of Classification Accuracy for "Noisy" Attributes in the Absence of Noise in a Learning Sample

Here we present the results of researching the effect of noise in a test samples on classification accuracy. First, the noise model "the absence of an attribute value" has been used. To restore absent values, the method of replacement by the average (*MORM)* and the method of *k nearest neighbours* (*kNN*) search were used. The problem of choosing *k* in the restoring algorithm was considered in [8] in detail. The situations with the noise presence in 5%, 10%, 20%, 30% in a chosen attribute were considered. The results of experiments with algorithms C4.5 and CN2 are given in tables 1 and 2 accordingly.

In table cells where the classification accuracy is represented, maximal values of the classification accuracy are marked by the bold type.

Table 1. Classification results of examples with noise "absent values" for the algorithm C 4.5

Data set	**Method of handling noise**	**Classification accuracy of "noisy" examples, %**					
		No noise	**Noise 5%**	**Noise 10%**	**Noise 20%**	**Noise 30%**	**Average**
HEART	*MORM*	75,31	75,8	75,31	75,06	75,06	**75,31**
	kNN		75,50	75,41	74,62	74,32	75,13
GLASS	*MORM*	72,31	72,31	72,31	72,00	71,69	72,08
	kNN		72,37	72,69	72,06	71,94	**72,27**
IRIS	*MORM*	95,56	95,56	96,00	96,02	97,34	96,23
	kNN		95,38	95,38	95,67	95,91	**95,59**
DIABETES	*MORM*	71	70,91	71,17	71,34	71,43	**71,21**
	kNN		71,22	71,17	71,05	71,15	71,15
MONKS1	*MORM*	81,71	81,94	81,94	82,64	83,1	**82,40**
	kNN		81,99	82,04	81,88	82,11	82,01
MONKS3	*MORM*	94,44	94,21	93,75	93,75	93,29	93,75
	kNN		94,03	93,94	93,75	93,56	**93,82**
CREDIT	*MORM*	78,06	78,67	79,49	79,28	79,49	**79,23**
	kNN		78,12	78,06	78,79	79,24	78,55
TIC-TAC-TOE	*MORM*	88,89	89,17	89,03	89,1	88,96	**89,06**
	kNN		88,83	88,81	88,82	88,51	88,74
WINE	*MORM*	90,74	90,74	90,37	89,63	89,63	90,09
	kNN		90,74	90,59	90,52	90,52	**90,59**
BREAST	*MORM*	97,07	97,07	97,36	97,56	97,76	**97,44**
	kNN		97,09	97,09	97,21	97,19	97,15

Table 2. Classification results of examples with noise “absent values” for the algorithm CN2

Data set	**Method of handling noise**	**Classification accuracy of “noisy” examples, %**					
		No noise	**Noise 5%**	**Noise 10%**	**Noise 20%**	**Noise 30%**	**Average**
HEART	MORM	74,07	74,56	74,57	74,81	75,55	**74,87**
	kNN		74,17	74,12	74,42	74,91	74,40
GLASS	MORM	66,15	66,15	66,15	66,15	66,46	66,23
	kNN		66,15	66,09	66,21	66,52	**66,24**
IRIS	MORM	95,56	95,56	95,56	96,0	95,56	95,67
	kNN		95,56	95,74	95,74	96,18	**95,80**
DIABETES	MORM	74,03	74,2	73,85	73,94	73,85	**73,96**
	kNN		74,01	73,84	74,044	73,89	73,94
MONKS1	MORM	100	100	100	100	100	**100**
	kNN		100	99,95	100	99,82	99,94
MONKS2	MORM	75,46	75,23	74,15	73,61	72,69	73,92
	kNN		75,05	74,54	74,03	73,06	**74,17**
MONKS3	MORM	91,2	91,2	91,44	90,97	91,2	**91,20**
	kNN		91,11	91,21	90,83	90,55	90,93
CREDIT	MORM	80,61	80,92	81,02	81,94	82,24	**81,53**
	kNN		80,73	80,75	80,92	81,43	80,96
TIC-TAC-TOE	MORM	93,75	93,75	94,1	94,16	94,44	**94,11**
	kNN		93,62	93,42	93,39	93,06	93,37
WINE	MORM	96,3	96,67	96,3	96,67	96,3	96,48
	kNN		96,37	96,44	96,3	96,15	**96,31**
BREAST	MORM	94,63	94,63	94,44	94,34	94,15	94,39
	kNN		94,65	94,53	94,55	94,46	**94,55**

On the whole, one can make the conclusion that the restoring algorithm based on nearest neighbours (*kNN*) search is more efficient than *MORM*.

While testing noise of "permutation" and "scrambling” types, we learnt that algorithms C4.5 and CN2 work quite well with handling "scrambling" noise when some attribute values are replaced by erroneous but the common value distribution of an informative attribute is retained. In this case classification accuracy reduced by 2% at growth of a noise level up to 20% on the average. This noise model is more characteristic for real DBs. Experiment results are illustrated in tables 3 and 4.

Table 3. Classification accuracy of examples with noise for the algorithm C 4.5

Data sets	Type/Noise level	Noise 0%	Noise 5%	Noise 10%	Noise 15%	Noise 20%	Average
HEART	Permutation	**77,78**	77,28	76,54	75,06	75,55	76,12
	Scrambling		77,53	78,27	77,53	76,79	**77,53**
GLASS	Permutation	**63,08**	62,77	60,92	60,62	60	61,08
	Scrambling		62,77	63,08	62,77	62,16	**62,70**
IRIS	Permutation	**93,33**	91,55	86,67	84,89	82,67	86,45
	Scrambling		92	88,44	87,55	86,67	**88,67**
DIABETES	Permutation	**67,97**	67,71	67,10	66,32	65,28	66,60
	Scrambling		67,88	66,5	67,27	66,93	**67,14**
MONKS1	Permutation	**81,71**	80,56	78,47	78,24	79,17	79,11
	Scrambling		81,02	78,70	79,63	78,94	**79,57**
MONKS3	Permutation	**94,44**	91,67	90,28	88,89	87,96	89,7
	Scrambling		94,21	94,44	94,44	93,98	**94,27**
CREDIT	Permutation	**84,18**	81,32	80,71	75,71	74,90	78,16
	Scrambling		82,24	82,85	81,02	79,37	**81,37**
TIC-TAC-TOE	Permutation	**84,03**	83,12	80,42	80,69	78,40	80,66
	Scrambling		82,78	82,22	81,87	80,21	**81,77**
WINE	Permutation	**81,48**	80	78,518	77,41	73,33	77,31
	Scrambling		80	80,37	77,41	79,63	**79,35**
BREAST	Permutation	**94,15**	91,71	89,27	87,80	83,03	87,95
	Scrambling		93,17	90,54	91,42	89,75	**91,22**

Table 4. Classification accuracy of examples with noise for the algorithm CN2

Data sets	Type/Noise level	Noise 0%	Noise 5%	Noise 10%	Noise 15%	Noise 20%	Average
HEART	Permutation	**77,78**	77,28	76,29	75,80	74,57	75,99
	Scrambling		77,53	77,04	77,04	77,28	**77,22**
IRIS	Permutation	**88,89**	88,00	86,22	82,66	82,22	84,78
	Scrambling		86,67	88,00	86,22	84,44	**86,33**
DIABETES	Permutation	**66,67**	66,75	65,37	65,71	64,33	65,54
	Scrambling		67,88	66,50	66,93	65,63	**66,73**
MONKS1	Permutation	**100**	99,07	96,99	95,37	94,44	96,47
	Scrambling		99,54	97,69	98,15	97,69	**98,27**
MONKS3	Permutation	**91,2**	90,05	87,96	85,65	85,19	87,21
	Scrambling		91,44	91,44	91,2	91,90	**91,50**

Table 4. (*continued*)

CREDIT	Permutation	**84,69**	82,75	80,93	79,59	76,63	79,97
	Scrambling		83,98	82,96	82,34	81,34	**82,66**
TIC-TAC-TOE	Permutation	**90,62**	89,09	87,85	86,74	85,35	87,26
	Scrambling		89,93	89,794	87,98	88,40	**89,03**
WINE	Permutation	**92,59**	91,85	92,59	92,22	91,85	92,13
	Scrambling		91,85	91,85	92,22	92,96	**92,22**
BREAST	Permutation	**90,73**	90,63	90,44	90,93	89,46	90,36
	Scrambling		90,24	90,34	90,44	90,15	**90,29**

8.3 Assessment of Classification Accuracy for "Noisy" Examples in the Presence of Noise in a Learning Sample

In the given experiments, "absent value" noise is artificially introduced in a learning sample. By using the restoring procedures such as *kNN*, *MORM*, internal procedures of algorithms C4.5 and CN2 as well as deleting "noisy" examples, generalized notions are built. Then for decision rules being built, the examination on data of test sample without noise is produced.

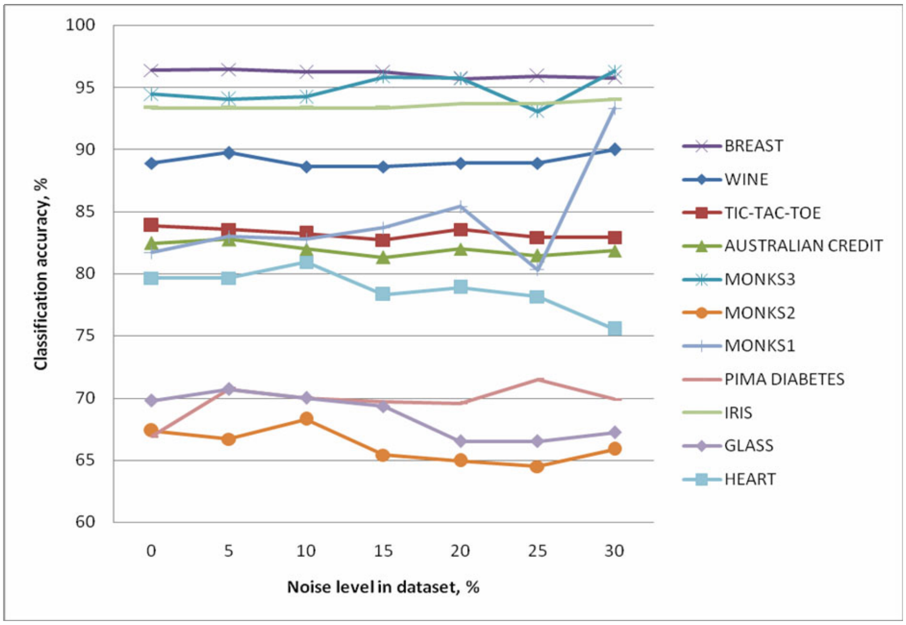


Fig. 1. Dependence of the classification accuracy on noise level in a learning sample for the algorithm C4.5

In fig.1 and 2 the test results are presented from which we can see that under introducing noise up to 30% in a learning sample, the quality of formed rules is not diminished and even in some cases the classification accuracy of new objects is higher than in absence (0%) of noise in a learning sample. By using the algorithm C 4.5 it is true for all samples excluding Monks 2, Glass, Heart. For the given samples, reduction of the level of quality of formed rules (not more than by 4%) was observed but it is worth to note that noise level was high enough (more than 10%).

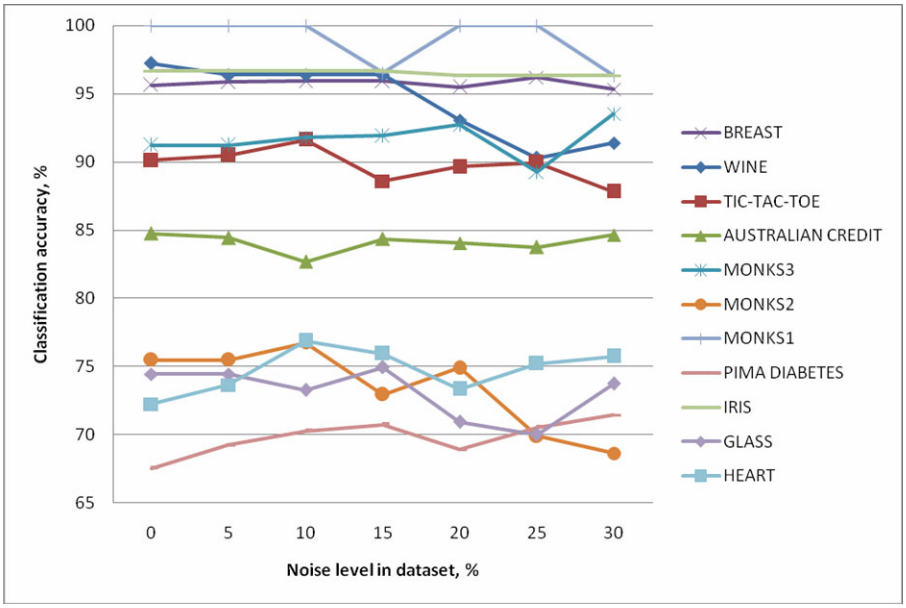


Fig. 2. Dependence of the classification accuracy on noise level in a learning sample for the algorithm CN2

For the algorithm CN2 a similar situation arises practically for all data sets. This dependence is, undoubtedly, true for samples *Iris, Breast, Heart, Diabetes.* For the rest cases with increasing a noise level, increasing erroneous classification level of test sample objects takes place. However, as well as for the C 4.5 algorithm, it appears at a significant quantity of "noisy" examples.

9 Conclusions

We considered the problem of information generalization and examined the ways of its solution in the presence of noise in original data.

The noise models in DB tables, the results of which are the absence of attribute values or the distortion and scrambling of attribute values in a learning sample, have been considered. The algorithm IDTUV2 allowing to process learning samples, containing examples with noisy values, have been suggested. The system of building generalized concepts using the obtained theoretical results has been developed and programmly implemented.

The obtained results of modeling have shown that the algorithms C4.5 и CN2 in the combination with the restoring algorithm allow to handle data efficiency in the presence of noise of a different type.

The paper has been written with the financial support of RFBR (No. 08-07-00212a, 09-01-00076a).

References

1. Vagin, V.N., Ju, G.E., Zagoryanskaya, A.A., Fomina, M.V.: Exact and Plausible Inference in Intelligent Systems. In: Vagin, V.N., Pospelov, D.A. (eds.), 2nd edn., p. 716 (2008) (in Russian)
2. Quinlan, J.R.: Induction of Decision Trees. Machine Learning 1, 81–106 (1986)
3. Quinlan, J.R.: Improved Use of Continuous Attributes in C 4.5. Journal of Artifical Intelligence Research 4, 77–90 (1996)
4. Clark, P., Boswell, R.: Rule Induction with CN2, pp. 151–163 (1991)
5. Clark, P., Niblett, T.: The CN2 Induction Algorithm. Machine Learning 3, 261–283 (1989)
6. Mookerjee, V., Mannino, M., Gilson, R.: Improving the Performance Stability of Inductive Expert Systems under Input Noise. Information Systems Research 6(4), 328–356 (1995)
7. Berisha, A.M., Vagin, V.N., Kulikov, A.V., Fomina, M.V.: Methods of Knowledge Discovery in "Noisy" Databases. Journal of Computer and Systems Sciences International 44(6), 973–987 (2005)
8. Vagin, V.N., Fomina, M.V., Kulikov, A.V.: The Problem of Object Recognition in the Presence of Noise in Original Data. In: 10th Scandinavian Conference on Artificial Intelligence, SCAI 2008, pp. 60–67 (2008)
9. Merz, C.J., Murphy, P.M.: UCI Repository of Machine Learning Datasets. Information and Computer Science University of California, Irvine, CA 92697-3425 (1998), `http://archive.ics.uci.edu/ml/`

Object Class Recognition Using SIFT and Bayesian Networks

Leonardo Chang[1,2], Miriam Monica Duarte[2],
Luis Enrique Sucar[2], and Eduardo F. Morales[2]

[1] Advanced Technologies Application Center,
7a # 21812 b/ 218 y 222, Siboney, Playa, P.C. 12200, Havana, Cuba
lchang@cenatav.co.cu

[2] National Institute for Astrophysics, Optics and Electronics,
Luis Enrique Erro # 1, Tonantzintla, Puebla, Mexico
{lchang,mduarte,esucar,emorales}@ccc.inaoep.mx

Abstract. Several methods have been presented in the literature that successfully used SIFT features for object identification, as they are reasonably invariant to translation, rotation, scale, illumination and partial occlusion. However, they have poor performance for classification tasks. In this work, SIFT features are used to solve problems of object class recognition in images using a two-step process. In its first step, the proposed method performs clustering on the extracted features in order to characterize the appearance of classes. Then, in the classification step, it uses a three layer Bayesian network for object class recognition. Experiments show quantitatively that clusters of SIFT features are suitable to represent classes of objects. The main contributions of this paper are the introduction of a Bayesian network approach in the classification step to improve performance in an object class recognition task, and a detailed experimentation that shows robustness to changes in illumination, scale, rotation and partial occlusion.

Keywords: Object class recognition, local features, SIFT, clustering, Bayesian networks.

1 Introduction

In the last few years, local features have proven to be very effective in finding distinctive features between different views of a scene. The traditional idea of these methods is to first identify structures or significant points in the image and to obtain a discriminant description of these structures from its surroundings, which is then used for comparison using a similarity measure between these descriptors. A keypoint detector is designed to find the same point in different images even if the point is in different locations and scales. Different methods have been proposed in the literature. A study and comparison of these approaches is presented in [11].

One of the most popular and widely used local approach is the SIFT (Scale Invariant Features Transform) method, proposed by Lowe [7]. The features extracted by SIFT are largely invariant to scale, rotation, illumination changes, noise and small changes in viewing direction. The SIFT descriptors have shown better results than other local descriptors [10].

The SIFT and local features have been mainly used for the identification of particular objects within a scene. For instance, a particular book is given to a system, which extracts its SIFT features and uses them to recognize that particular book. However, such features cannot be used to recognize another book or books in general on the scene.

In this work we use SIFT features to recognize object classes (e.g., books) in order to provide robustness to changes in scale, rotation, illumination and partial occlusion. The proposed method, in the training phase, performs clustering on the features extracted from the training set. Each feature in each cluster is labeled with its corresponding class in order to characterize the appearance of object classes. In the classification step, for a new image, the SIFT features are extracted, and for each feature the cluster from the learned model to which it belongs is identified. Information from the identified clusters is then used to find the most probable class. To represent this idea, we introduce the use of a three layer Bayesian network. Three experiments were conducted to test the performance of the proposed method. These experiments showed quantitatively that the use of SIFT local features, clustering and Bayesian networks are suitable to represent and recognize object classes. They also showed the invariance of the method in the presence of changes in illumination, scale, rotation and partial occlusion.

The main contributions of this paper are the following. Firstly, we introduce a Bayesian network approach in the classification step to improve performance on this stage. Secondly, we show that clustering over local features provides robustness to changes in illumination, scale, rotation and partial occlusion. We also show that this kind of approach outperforms a straightforward classification method using SIFT features. These last two issues are mentioned in the literature but there is no detailed experimental evidence to support them.

2 Related Work

Most objects class recognition methods characterize objects by their global appearance, usually of the entire image. These methods are not robust to occlusion or variations such as rotation or scale. Moreover, these methods are only applicable to rigid objects. Local features have become very popular to give solution to the limitations of these methods in object detection and recognition.

For object class recognition, many methods use clustering as an intermediate level of representation [1][6]. Due to the robustness of local features and the good results of clustering in objects classification, several authors have recently been investigating the use of clustering for object class recognition using local features based approaches. In [2], for invariant region detection, the authors use the Harris-Laplace [9] and the Kadir and Brady [5] detector. These regions are described using the SIFT descriptor [7]. In their work, Dorkó and Schmid perform clustering of descriptors to characterize class appearance. Then, they build classifiers of smaller parts of objects from the clusters formed. By discarding several of these clusters they kept only the most discriminative ones.

In [8], Mikolajczyk *et al.* evaluate the performance of various methods based on local features in the object class recognition task. The invariant region detectors evaluated were Harris-Laplace, SIFT, Hessian-Laplace, and MSER. The evaluated features descriptors were SIFT, GLOH, SIFT-PCA, Moments, and Cross-Correlation. In their paper the authors evaluate several detector-descriptor combinations. Clustering is also performed on the descriptors to characterize the appearance of classes. To classify a new sample, the extracted descriptors are matched with the clusters obtained and a threshold determines the class membership.

In these works it is mentioned that their proposed methods have invariance to occlusion, changes in illumination, rotation and scale. However, there is no experimentation for the above, neither do they express how robust these methods are. It is also assumed that their proposed methods outperform a straightforward classification method using local features, but no evidence of this is given. In this paper, we analyze these facts through a set of detailed experiments over our proposed method.

The method proposed in this work also performs clustering on the descriptors of the features extracted from training images. The main difference with the previous mentioned methods is the use of a Bayesian network in the classification stage in order to improve performance on this stage. A deeper experimentation to measure the behavior against changes in illumination, scale, viewpoint and partial occlusion is presented as well. It is also shown how the use of clustering and Bayesian networks outperforms the traditional use of local features in object class recognition tasks.

3 SIFT Features Descriptors

SIFT is one the most widely used local approaches. It finds local structures that are present in different views of the image. It also provides a description of these structures reasonably invariant to image variations such as translation, rotation, scale, illumination and affine transformations. Moreover, several studies have shown that the SIFT descriptor performs better than others.

The first stages of the SIFT algorithm find the coordinates of keypoints in a certain scale and assign an orientation to each one. The results of these steps guarantee invariance to image location, scale and rotation. Then, a descriptor is computed for each keypoint. This descriptor must be highly distinctive and partially robust to other variations such as illumination and 3D viewpoint.

To create the descriptor, Lowe proposed an array of 4×4 histograms of 8 bins [7]. These histograms are calculated from the values of orientation and magnitude of the gradient in a region of 16×16 pixels around the point so that each histogram is formed from a subregion of 4×4. The descriptor vector is a result of the concatenation of these histograms. Since there are $4 \times 4 = 16$ histograms of 8 bins each, the resulting vector is of size 128. This vector is normalized in order to achieve invariance to illumination changes.

The distinctiveness of these descriptors allows us to use a simple algorithm to compare the collected set of feature vectors from one image to another in order to find correspondences between feature points in each image. These correspondences are adequate to identify particular objects in the image, but not to identify object classes. With this purpose in mind, in this paper, SIFT feature descriptors are clustered to characterize object classes and are incorporated in a Bayesian network classifier.

4 Learning Object Classes

A model able to generalize beyond each object in the training set and that allows us to learn a general structure of each class is desired. Moreover, learning should be possible from a small number of samples. With this aim and in accordance with several studies reported in the literature (mentioned in Section 2), clustering is performed on feature descriptors extracted from the training images.

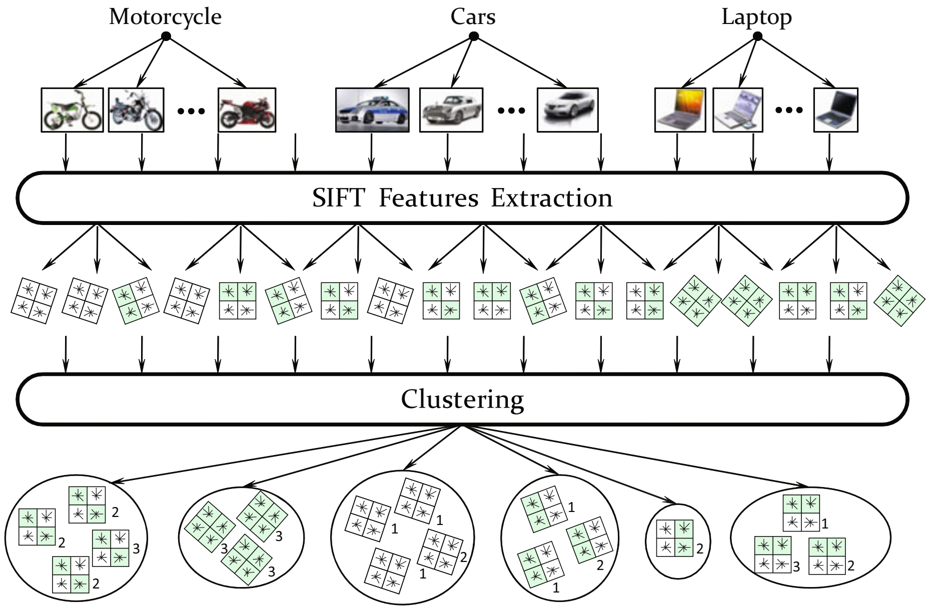


Fig. 1. SIFT local features are extracted from the training set formed by several sample images per class. Later, features descriptors are clustered and each feature in each cluster is labeled with its corresponding class.

Figure 1 shows a high level diagram of the class learning method used, which is summarized as follows:

1. For each training image, SIFT local features are extracted.
2. Then, clustering is performed over the features descriptors.
3. Finally, each descriptor in each cluster is labeled with its corresponding class.

Clusters are expected to have high accuracy i.e., each cluster is representative of only one class. In practice, this not always occurs so there could be clusters that are shared by several classes. Additional methods will be needed in the classification stage to solve these ambiguities.

4.1 SIFT Features Clustering

To build clusters of descriptors, the agglomerative hierarchical clustering method proposed by [4] is used. Unlike K-means or EM-clustering, this algorithm does not depend on initialization. Furthermore, it has been reported superior to K-means [3].

Given F features descriptors extracted from all the images in the training set, the clustering is initialized with F clusters, each one containing one descriptor only. In each iteration the two clusters with the highest cohesion are merged.

The similarity between any two clusters can be measured in several ways, the most common are single linkage, complete linkage and average linkage. In this paper, average linkage is used, which is defined as the average distance of every element in a cluster to any other element in other cluster:

$$D(k,l) = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} d(k_m, l_n), \tag{1}$$

where M and N are the number of descriptors in the clusters k and l respectively.

Agglomerative clustering produces a hierarchy of associations of clusters until the cut off criterion halts the process. Therefore, after each iteration, a new cluster is obtained from the pair of clusters with the highest similarity above a given value. This value is used as cut off criterion.

5 Recognizing Object Classes

Given a new sample image, classification is performed by first extracting the SIFT features from the input image. Then, for each of these features, a cluster is associated from the learned model and finally, from this instantiation of the model, the class of the input object is determined. Figure 2 shows a layout of the proposed method.

This idea can be represented as a three layer Bayesian network (BN). The graphical representation of this BN is shown in Figure 3. At the first layer we have the trained object classes represented by $c_1, c_2, ..., c_C$ where C is the number of classes. At the second layer, clusters obtained in the training phase are represented by $k_1, k_2, ..., k_K$ where K is the number of obtained clusters. Finally, the third layer represents the features extracted from the new object, and are represented by the nodes $f_1, f_2, ..., f_F$ where F is the number of features extracted from the image.

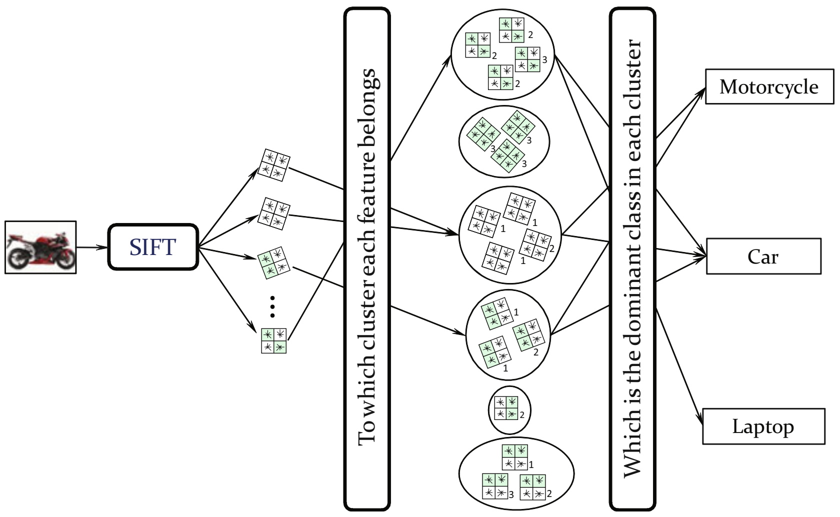

SIFT
To which cluster each feature belongs
Which is the dominant class in each cluster
Motorcycle
Car
Laptop

Fig. 2. Classification scheme for a new image. SIFT features are extracted from this image and for each feature the cluster from the learned model to which it belongs is identified. The object class is the majority class in these clusters.

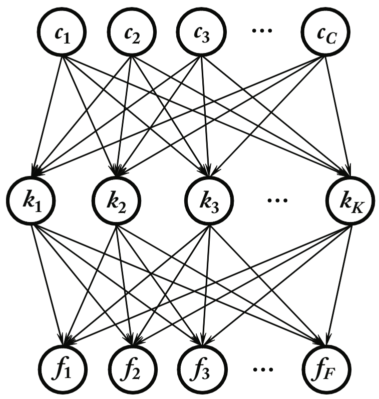


Fig. 3. Graphical representation of the three layer Bayesian network used to classify a new object

Using this model, the classification of a new image I is performed as follows:

1. SIFT features are extracted from the input image I.
2. For each feature f extracted from I, cluster k_f to which it belongs is obtained. The cluster with the highest membership probability of the feature f is selected. This probability is function of the distance between the cluster

and the feature, which is normalized by the distance between the two most distant clusters. The same distance D defined in Equation 1 is used:

$$k_f = \arg\max_i P(f|k_i)P(k_i), \text{ where}$$

$$P(f|k_i) = 1 - \frac{D(f,k_i)}{\max_{kl} D(k_k,k_l)}$$

3. For each cluster $k_{f_1}, k_{f_2}, ..., k_{f_F}$ selected in the previous step (note that more than one feature could be in the same cluster), the probability of each class given this evidence is obtained, this probability is extracted from the trained model, propagating further the probability obtained in step 2.
4. Finally, the object class is the one whose sum of occurrence probabilities given each cluster selected in step 2 is the highest:

$$c^* = \arg\max_i \sum_f P(C_i|k_f)P(k_f)$$

6 Experiments and Results

This section presents a quantitative evaluation of the proposed method and discusses the main results obtained.

For the conducted experiments, images from the Pascal[1] collection were used. This database contains 101 different classes of objects and different numbers of images per class, the format is JPG and the average size is 300 × 300 pixels. Each image contains only one object centered in the image.

In order to test the performance of the proposed method, a system was trained to recognize four classes of objects (i.e., camera, dollar bill, motorcycle, and wristwatch), which were randomly selected. For training, 20 images per class were used, also randomly selected. Example images from the training set are shown in Figure 4.

Three experiments were conducted to evaluate the proposed method. The goal of the first experiment is to measure the performance of the proposed method in normal conditions (i.e., illumination, occlusion, rotation and scale problems-free images). The second experiment compares the method proposed in this paper with a straightforward classification method also using SIFT features. Finally, the third experiment measures how the performance of the proposed method behaves in the presence of partial occlusion and variation in illumination, scale and rotation in the test set.

The performance indicators used were recall, precision, true negative rate and accuracy. The recall rate measures the proportion of actual positives which are correctly identified as such ($recall = TP/(TP + FN)$). Precision is defined as the proportion of the true positives against all the positive results ($precision = TP/(TP + FP)$). The True Negative Rate (TNR) measures the proportion of negatives which are correctly identified ($TNR = TN/(FP+TN)$). The accuracy is the proportion of true results, both true positives and true negative, in the population ($accuracy = (TP + TN)/(P + N)$).

[1] Available online at: "http://pascallin.ecs.soton.ac.uk/challenges/VOC/download/101objects.tar.gz"

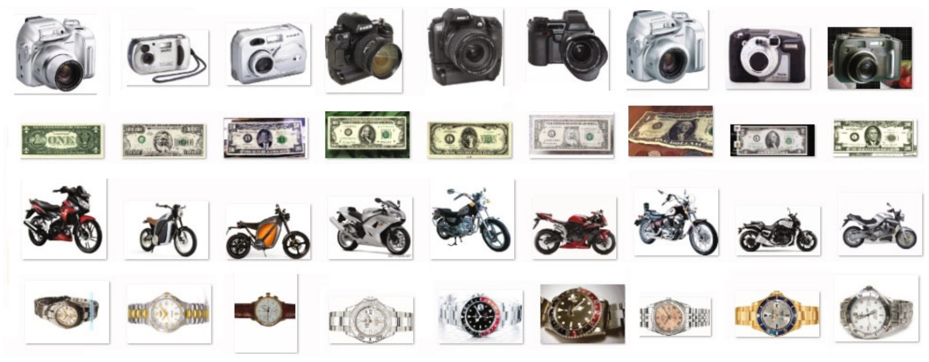


Fig. 4. Example images from the training set. The training set is composed of 20 images for each of the 4 classes. These images were randomly selected from the database.

6.1 Experiment 1

In Experiment 1, results were obtained for 100 test images per class. These images have small variations in occlusion, scale, illumination and rotation. Images from the training set were not in the test set. Table 1 shows the results obtained in Experiment 1.

Table 1. Performance indicators for Experiment 1

	Recall (%)	Precision (%)	TrueNegativeRate (%)	Accuracy (%)
Camera	84.0	94.6	98.3	93.5
Dollar bill	100	89.2	96.0	95.0
Motorcycle	99.0	90.5	96.7	95.0
Wristwatch	89.0	98.9	99.7	94.5
Average	90.7	93.3	97.6	94.5

As could be seen in Table 1, all the measures averages were over 90%, which indicates the high performance of the proposed method.

6.2 Experiment 2

In order to evaluate the improvement introduced by the clustering of SIFT descriptors on the representation of object classes and the use of a Bayesian network in the classification phase, in this section we compare the method proposed in this paper with a straightforward classification method also using SIFT features, which is taken as baseline. This method is summarized as follows:

1. Extract SIFT features of each image from the training set.
2. For a new image I extract its SIFT features.

3. This image is matched with each of the images of the training set. The matching method used is the one proposed by Lowe in [7].
4. The class of the input image will be the one that receives the highest number of correspondences with image I.

Table 2 shows a comparison between the results obtained by the baseline method and the results obtained in Experiment 1. To perform this experiment, the same training and test sets that in Experiments 1 were used.

Table 2. Comparison of Baseline and Experiment 1

	Baseline	Experiment 1
Recall (%)	68.0	90.7
Precision (%)	80.9	93.3
TrueNegativeRate (%)	89.3	97.6
Accuracy (%)	84.0	94.5

As could be noticed in Table 2, the proposed method surpassed in each of the performance measures to the baseline method by a wide margin. This result gives evidence of the improvement introduced by the clustering of SIFT descriptors on the representation of object classes and the use of a Bayesian network in the classification phase.

6.3 Experiment 3

The aim of Experiment 3 is to test the robustness of the proposed method to changes in illumination, occlusion, scale and rotation. For Experiment 3, 10

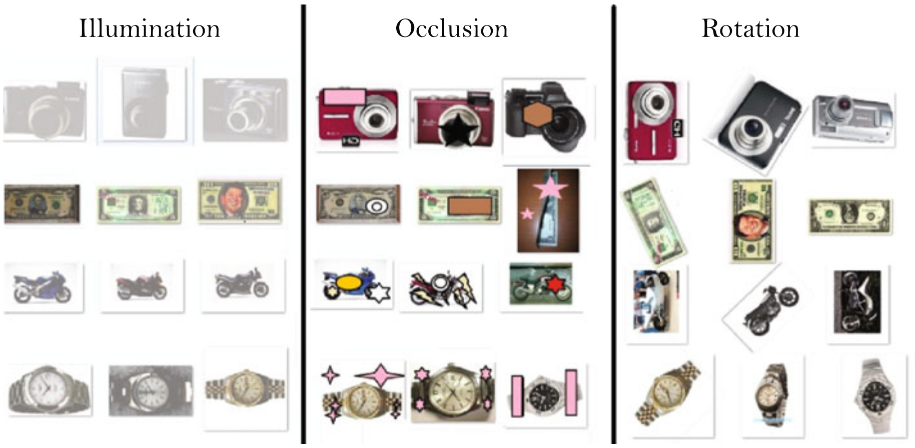


Fig. 5. Example images from the test set used for the Experiment 3. These images present partial occlusion and changes in illumination, rotation and scale.

Table 3. Performance indicators for Experiment 3

	Recall (%)	Precision (%)	TrueNegativeRate (%)	Accuracy (%)
Camera	94.8	92.0	97.5	96.5
Dollar bill	95.3	98.0	99.3	98.0
Motorcycle	92.5	94.0	97.9	96.5
Wristwatch	100	96.0	98.7	99.0
Average	95.6	95.0	98.3	97.5

Table 4. Recall and precision measures (%) for each type of image alteration in Experiment 3

	Occlusion		Illumination		Scale 2x		Scale 0.5x		Rotation	
	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision
Camera	100	90.0	100	70.0	90.9	100	100	100	83.3	100
Dollar bill	100	100	76.9	100	100	100	100	100	100	90.0
Motorcycle	90.9	100	81.8	90.0	100	90.0	100	100	90.0	90.0
Wristwatch	100	100	100	90.0	100	100	100	100	100	90.0

images that were correctly classified in Experiment 1 were randomly selected for each class. Variations in occlusion, scale, illumination and rotation were artificially introduced to each of these images, resulting in 40 images per class. Example images from the test set used in this experiment are shown in Figure 5.

Table 3 shows the performance results obtained in Experiment 3. As it could be seen, the average values of performance are maintained above 95%, showing the robustness of the proposed method to variations in illumination, occlusion, scale and rotation.

The recall and precision measures obtained for each kind of variation introduced to the test set is shown in Table 4. It could be noticed that there were no major falls in recall and precision rates, showing the largest variations (30 %) in the precision on the illumination changed images in the class camera.

7 Conclusions

As a result of this work, a method for recognizing object classes using SIFT features have been developed. The proposed method performs clustering on the descriptors of the detected points to characterize the appearance of object classes. It also introduces the use of a three layer Bayesian network in the classification stage to improve classification rates. Three experiments were conducted to evaluate the proposed method. They showed that SIFT features are suitable to represent object classes, and evidenced the improvement achieved by clustering SIFT descriptors and using a Bayesian network for classification. These experiments also showed quantitatively the invariance of the method to illumination changes, scale, rotation and occlusion. It also provided experimental evidence that supports that a method based on clustering of SIFT features outperforms a straightforward object recognition method to identify object classes.

As future work, the localization of objects in the image will be investigated, trying to learn the spatial relationships between the local features and clusters that describe an object class.

Acknowledgements

This project was supported in part by CONACYT grant No. 103876. L. Chang was supported in part by CONACYT scholarship No. 240251.

References

1. Agarwal, S., Awan, A., Roth, D.: Learning to detect objects in images via a sparse, part-based representation. IEEE Trans. Pattern Anal. Mach. Intell. 26(11), 1475–1490 (2004)
2. Dorkó, G., Schmid, C.: Object class recognition using discriminative local features. Technical report, IEEE Transactions on Pattern Analysis and Machine Intelligence (2005)
3. Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice-Hall, Inc., Upper Saddle River (1988)
4. Johnson, S.C.: Hierarchical clustering schemes. Psychometrika 2, 241–254 (1967)
5. Kadir, T., Brady, M.: Scale, saliency and image description. International Journal of Computer Vision 45(2), 83–105 (2001)
6. Leibe, B., Seemann, E., Schiele, B.: Pedestrian detection in crowded scenes. In: CVPR 2005: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), Washington, DC, USA, vol. 1, pp. 878–885. IEEE Computer Society, Los Alamitos (2005)
7. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
8. Mikolajczyk, K., Leibe, B., Schiele, B.: Local features for object class recognition. In: ICCV 2005: Proceedings of the Tenth IEEE International Conference on Computer Vision, Washington, DC, USA, pp. 1792–1799. IEEE Computer Society, Los Alamitos (2005)
9. Mikolajczyk, K., Schmid, C.: Indexing based on scale invariant interest points. In: Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada, pp. 525–531 (2001)
10. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Mach. Intell. 27(10), 1615–1630 (2005)
11. Tuytelaars, T., Mikolajczyk, K.: Local invariant feature detectors: A survey. Foundations and Trends in Computer Graphics and Vision 3(3), 177–280 (2007)

Boosting Based Conditional Quantile Estimation for Regression and Binary Classification

Songfeng Zheng

Department of Mathematics
Missouri State University, Springfield, MO 65897, USA
SongfengZheng@MissouriState.edu

Abstract. We introduce Quantile Boost (QBoost) algorithms which predict conditional quantiles of the interested response for regression and binary classification. Quantile Boost Regression (QBR) performs gradient descent in functional space to minimize the objective function used by quantile regression (QReg). In the classification scenario, the class label is defined via a hidden variable, and the quantiles of the class label are estimated by fitting the corresponding quantiles of the hidden variable. An equivalent form of the definition of quantile is introduced, whose smoothed version is employed as the objective function, which is maximized by gradient ascent in functional space to get the Quantile Boost Classification (QBC) algorithm. Extensive experiments show that QBoost performs better than the original QReg and other alternatives for regression and classification. Furthermore, QBoost is more robust to noisy predictors.

Keywords: Boosting, Quantile Regression, Classification.

1 Introduction

Least square regression aims to estimate the conditional expectation of the response Y given the predictor (vector) $\mathbf{x}$, i.e., $\mathrm{E}(Y|\mathbf{x})$. However, the mean value (or the conditional expectation) is sensitive to the outliers of the data [12]. Therefore, if the data is not homogeneously distributed, we expect the least square regression gives us a poor prediction.

The τ-th quantile of a distribution is defined as the value such that there is $100\tau\%$ of mass on the left side of it. Compared to the mean value, quantiles are more robust to outliers [12]. For a random variable Y, it can be proved [11] that

$$Q_\tau(Y) = \arg\min_c \mathrm{E}_Y[\rho_\tau(Y-c)],$$

where $Q_\tau(Y)$ is the τ-th quantile of Y, $\rho_\tau(r)$ is the "check function" [12] defined by

$$\rho_\tau(r) = rI(r \geq 0) - (1-\tau)r, \tag{1}$$

where $I(\cdot) = 1$ if the condition is true, otherwise $I(\cdot) = 0$.

Given data $\{(\mathbf{x}_i, Y_i), i = 1, \cdots, n\}$, with predictor $\mathbf{x}_i \in \mathbf{R}^d$ and response $Y_i \in \mathbf{R}$, let the τ-th conditional quantile of Y given $\mathbf{x}$ be $f(\mathbf{x})$. Similar to least square regression, quantile regression (QReg) [12] aims at estimating the conditional quantiles of the response given predictor vector $\mathbf{x}$ and can be summarized as

$$f^*(\cdot) = \arg\min_f \frac{1}{n} \sum_{i=1}^{n} \rho_\tau \left(Y_i - f(\mathbf{x}_i)\right), \tag{2}$$

which can be solved by linear programming algorithms [12] or MM algorithms [11]. However, when the predictor $\mathbf{x}$ is in high dimensional space, the aforementioned optimization methods for QReg might be inefficient. High dimension problems are ubiquitous in applications, e.g., image analysis, gene sequence analysis, to name a few. To the best of our knowledge, the problem of high dimensional predictor is not sufficiently addressed in quantile regression literature.

Motivated by the basic idea of gradient boosting algorithms [8], we propose to estimate the quantile regression function by minimizing the objective function in Eqn. (2) with functional gradient descent. In each step, we approximate the negative gradient of the objective function by a base function, and grow the model along that direction. This results Quantile Boost Regression (QBR) algorithm. In the binary classification scenario, we define the class label via a hidden variable, and the quantiles of the class label can be estimated by fitting the corresponding quantiles of the hidden variable. An equivalent form of the definition of quantile is introduced, whose smoothed version is employed as the objective function for classification. Similar to QBR, functional gradient ascent is applied to maximize the objective function, yielding the Quantile Boost Classification (QBC) algorithm. The obtained Quantile Boost (QBoost) algorithms are computationally efficient and converge to a local optimum, more importantly, they enable us to solve high dimensional problems efficiently.

The QBoost algorithms were tested extensively on publicly available datasets for regression and classification. On the regression experiments, QBR performs better than the original QReg in terms of check loss function. Moreover, the comparative experiment on noisy data indicates that QBR is more robust to noise. On classification problems, QBC was compared to binary QReg on a public dataset, the result shows that QBC performs better than binary QReg and is more robust to noisy predictors. On three high dimensional datasets from bioinformatics, binary QReg is not applicable due to its expensive computation, while QBC performs better than or similar to other alternatives in terms of 5 fold cross validation error rates. Furthermore, both QBC and QBR are able to select the most informative variables, inheriting the feature selection ability of boosting algorithm.

2 Boosting as Functional Gradient Descent

Boosting [7] is well known for its simplicity and good performance. The powerful feature selection mechanism of boosting makes it suitable to work in high

Algorithm 1. Generic Functional Gradient Descent

```
0: Initialize $f^{[0]}(\cdot)$ with
       $$f^{[0]}(\cdot) = \arg\min_c \frac{1}{n}\sum_{i=1}^{n} l(Y_i, c),$$
   or set $f^{[0]}(\cdot) = 0$, and set $m = 0$.
1: Increase $m$ by 1. Compute the negative gradient $-\frac{\partial}{\partial f} l(Y, f)$ and evaluate at $f^{[m-1]}(\mathbf{x}_i)$:
       $$U_i = -\frac{\partial l(Y_i, f)}{\partial f}\bigg|_{f=f^{[m-1]}(\mathbf{x}_i)}, \quad i = 1, \cdots, n.$$
2: Fit the negative gradient vector $U_1, \cdots, U_n$ to $\mathbf{x}_1, \cdots, \mathbf{x}_n$ by the base procedure (e.g. the weak learner in AdaBoost):
       $$\{(\mathbf{x}_i, U_i),\ i = 1, \cdots, n\} \longrightarrow g^{[m]}(\cdot).$$
3: Update the estimation by $f^{[m]}(\cdot) = f^{[m-1]}(\cdot) + \eta g^{[m]}(\cdot)$, where $\eta$ is a step-length factor.
4: Check the stopping criterion, if not satisfied, go to step 1.
```

dimensional spaces. Friedman et al. [8,9] developed a general statistical framework which yields a direct interpretation of boosting as a method for function estimate, which is a "stage-wise, additive model".

Consider the problem of function estimation

$$f^*(\mathbf{x}) = \arg\min_f \mathrm{E}[l(Y, f(\mathbf{x}))|\mathbf{x}],$$

where $l(\cdot,\cdot)$ is a loss function which is typically differentiable and convex with respect to the second argument. Estimating $f^*(\cdot)$ from the given data $\{(\mathbf{x}_i, Y_i), i = 1, \cdots, n\}$ can be performed by minimizing the empirical loss $n^{-1}\sum_{i=1}^{n} l(Y_i, f(\mathbf{x}_i))$ and pursuing iterative steepest descent in functional space. This leads us to the generic functional gradient descent algorithm [1,8] as shown in Algorithm 1.

Many boosting algorithms can be understood as functional gradient descent with appropriate loss function. For example, if we choose $l(Y, f) = \exp(-(2Y - 1)f)$, we would recover AdaBoost [9]; if $l(Y, f) = (Y - f)^2/2$ is used, we would result in L_2 Boost [2].

3 Quantile Boost for Regression

We consider the problem of estimating quantile regression function in the general framework of functional gradient descent with the loss function

$$l(y, f) = \rho_\tau(y - f) = (y - f)I(y - f \geq 0) - (1 - \tau)(y - f).$$

A direct application of Algorithm 1 yields the Quantile Boost Regression (QBR) algorithm, which is shown as Algorithm 2.

Algorithm 2. Quantile Boost Regression (QBR)

```
0: Given training data $\{(\mathbf{x}_i, Y_i),\ i = 1, \cdots, n\}$ and the desired quantile value $\tau$.
1: Initialize $f^{[0]}(\cdot)$ with
       $f^{[0]}(\cdot) = \tau\text{-th quantile of } (Y_1, \cdots, Y_n),$
   or set $f^{[0]}(\cdot) = 0$.
2: for $m = 1$ to $M$ do
3:     Compute the negative gradient $-\frac{\partial}{\partial f}\rho_\tau(Y - f)$ and evaluate at $f^{[m-1]}(\mathbf{x}_i)$:
           $U_i = I(Y_i - f^{[m-1]}(\mathbf{x}_i) \geq 0) - (1 - \tau).$
4:     Fit the negative gradient vector $U_1, \cdots, U_n$ to $\mathbf{x}_1, \cdots, \mathbf{x}_n$ by the base procedure
           $\{(\mathbf{x}_i, U_i),\ i = 1, \cdots, n\} \longrightarrow g^{[m]}(\cdot).$
5:     Update the estimation by $f^{[m]}(\cdot) = f^{[m-1]}(\cdot) + \eta g^{[m]}(\cdot)$, where $\eta$ is a step-length
       factor.
6: end for
7: Output the estimated $\tau$-th quantile function $f^{[M]}(\mathbf{x})$.
```

Similar to [8], let the base procedure be $h(\mathbf{x}, \mathbf{a})$, where $\mathbf{a}$ is a parameter vector. Then the fourth step can be performed by an ordinary least square regression:

$$\mathbf{a}_m = \arg\min_{\mathbf{a}} \sum_{i=1}^{n} \left[U_i - h(\mathbf{x}_i, \mathbf{a})\right]^2,$$

hence the function $g^{[m]}(\mathbf{x}) = h(\mathbf{x}, \mathbf{a}_m)$ can be regarded as an approximation of the negative gradient by the base procedure. In step 5, the step-length factor η can be determined by line search

$$\eta = \arg\min_{\gamma} \sum_{i=1}^{n} \rho_\tau \left[Y_i - f^{[m-1]}(\mathbf{x}_i) - \gamma g^{[m]}(\mathbf{x}_i)\right].$$

However, line search algorithm is often time consuming, instead, in each iteration, we update the fitted function $f^{[m-1]}(\cdot)$ by a fixed but small step along the negative gradient direction. To guarantee the performance of the resulting model, we fix η at a small value as suggested by [1,8]. Similar to AdaBoost, QBR enables us to select most informative predictors if appropriate base learner is employed, and this will be demonstrated experimentally in Section 5.1.

There is a large volume of literature applying boosting to regression problems, for example, in [5,7,18], among others. However, all these methods estimate mean value of the response, not quantiles. Langford et al. [15] proposed to use classification technique in estimating the conditional quantile. For each given quantile value, their method trains a set of classifiers $\{c_t\}$ for a series of $t \in [0, 1]$, and the testing stage calculates the average of the outputs of the classifiers. Therefore, compared to the proposed QBR algorithm, this method is time consuming. Furthermore, it is not clear how to perform variable selection using the method in [15].

4 Quantile Boost for Classification

This section generalizes the QBR algorithm for classification problems.

4.1 Predicting Quantiles for Classification

Consider the following model:

$$Y^* = h(\mathbf{x}) + \epsilon \quad \text{and} \quad Y = I\{Y^* \geq 0\},$$

where Y^* is a continuous hidden variable, $h(\cdot)$ is the true model for Y^*, ϵ is a disturb, and Y is the observed label of the data. Let $Q_\tau(Y^*|\mathbf{x})$ be the τ-th conditional quantile of Y^* given $\mathbf{x}$, and let $g(\cdot)$ be a real nondecreasing function. Clearly

$$P(Y^* \geq y|\mathbf{x}) = P\left(g(Y^*) \geq g(y)|\mathbf{x}\right),$$

it follows that

$$g\left(Q_\tau(Y^*|\mathbf{x})\right) = Q_\tau(g(Y^*)|\mathbf{x}). \tag{3}$$

Since the indicator function $I(t \geq 0)$ is nondecreasing w.r.t. t, by Eqn. (3), we have

$$I\left(Q_\tau(Y^*|\mathbf{x}) \geq 0\right) = Q_\tau(I(Y^* \geq 0)|\mathbf{x}) = Q_\tau(Y|\mathbf{x}),$$

that is, the conditional quantile function of Y can be obtained by fitting the corresponding conditional quantile of Y^*. If we model the τ-th conditional quantile of the latent variable Y^* by $f(\mathbf{x}, \boldsymbol{\beta})$ with $\boldsymbol{\beta}$ as the parameter vector, i.e.

$$Q_\tau(Y^*|\mathbf{x}) = f(\mathbf{x}, \boldsymbol{\beta}), \tag{4}$$

it follows that the conditional quantile of the binary variable Y can be modeled as

$$Q_\tau(Y|\mathbf{x}) = I\left(f(\mathbf{x}, \boldsymbol{\beta}) \geq 0\right). \tag{5}$$

From the relation $Y = I(Y^* \geq 0)$, it follows that

$$P(Y = 1|\mathbf{x}) = P\left(Y^* \geq 0|\mathbf{x}\right), \tag{6}$$

while Eqn. (4) implies that

$$P(Y^* \geq f(\mathbf{x}, \boldsymbol{\beta})|\mathbf{x}) = 1 - \tau. \tag{7}$$

Thus, if $f(\mathbf{x}, \boldsymbol{\beta}) = 0$, combining Eqn. (6) and (7) yields

$$P(Y = 1|\mathbf{x}) = P\left(Y^* \geq 0|\mathbf{x}\right) = P\left(Y^* \geq f(\mathbf{x}, \boldsymbol{\beta})|\mathbf{x}\right) = 1 - \tau;$$

if $f(\mathbf{x}, \boldsymbol{\beta}) > 0$,

$$P(Y = 1|\mathbf{x}) = P\left(Y^* \geq 0|\mathbf{x}\right) > P\left(Y^* \geq f(\mathbf{x}, \boldsymbol{\beta})|\mathbf{x}\right) = 1 - \tau.$$

In summary, we have the following relation:

$$P\left(Y = 1 \,\middle|\, f(\mathbf{x}, \boldsymbol{\beta}) \gtreqless 0\right) \gtreqless 1 - \tau, \tag{8}$$

which is an inequality of the posterior probability of response given the predictor vector. Hence, if we make decision with cut-off posterior probability $1 - \tau$, we need to fit a quantile regression model with quantile value τ. Once the model is fit, i.e., the parameter vector $\boldsymbol{\beta}$ is estimated as $\mathbf{b}$, we can make prediction by

$$\hat{Y} = I(f(\mathbf{x}, \mathbf{b}) \geq 0),$$

where $\hat{Y}$ is the predicted label for the predictor vector $\mathbf{x}$.

4.2 Quantile Boost for Classification

To fit the model for the τ-th quantile of Y, i.e., to estimate the parameter vector $\boldsymbol{\beta}$ in Eqn. (5), similar to QBR, the estimated parameter vector $\mathbf{b}$ can be obtained by solving:

$$\mathbf{b} = \arg\min_{\boldsymbol{\beta}} \left\{ L(\boldsymbol{\beta}) = \sum_{i=1}^{n} \rho_\tau \left[Y_i - I(f(\mathbf{x}, \boldsymbol{\beta}) \geq 0) \right] \right\}.$$

We can show (details are omitted due to space limit) that the above minimization problem is equivalent to the following maximization problem [14]:

$$\mathbf{b} = \arg\max_{\boldsymbol{\beta}} \left\{ S(\boldsymbol{\beta}) = \sum_{i=1}^{n} [Y_i - (1 - \tau)] I(f(\mathbf{x}, \boldsymbol{\beta}) \geq 0) \right\}. \tag{9}$$

However, the function $S(\boldsymbol{\beta})$ is not differentiable. To apply gradient based optimization methods, we replace $I(f(\mathbf{x}, \boldsymbol{\beta}) \geq 0)$ by its smoothed version, solving

$$\mathbf{b} = \arg\max_{\boldsymbol{\beta}} \left\{ S(\boldsymbol{\beta}, h) = \sum_{i=1}^{n} [Y_i - (1 - \tau)] K\left(\frac{f(\mathbf{x}, \boldsymbol{\beta})}{h}\right) \right\}, \tag{10}$$

where h is a small positive value, and $K(\cdot)$ is smoothed version of the indicator function $I(t \geq 0)$ with the following properties:

$$K(t) > 0,\ \forall t \in \mathbf{R}, \quad \lim_{t \to \infty} K(t) = 1, \quad \lim_{t \to -\infty} K(t) = 0.$$

In this paper, we take $K(\cdot)$ as the standard normal cumulative distribution function

$$\Phi(z) = \int_{-\infty}^{z} \varphi(t)dt \quad \text{with} \quad \varphi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}. \tag{11}$$

Let each term in the objective function (10) be

$$l(Y, f) = [Y - (1 - \tau)]K(f/h).$$

Following the general steps of functional gradient ascent, we obtain the Quantile Boost Classification (QBC) algorithm as sketched in Algorithm 3. Similar to QBR, the parameter η in QBC can be fixed at a small value instead of performing a line search.

Algorithm 3. Quantile Boost Classification (QBC)

```
0: Given training data $\{(\mathbf{x}_i, Y_i), i = 1, \cdots, n\}$ with $Y_i \in \{0, 1\}$, and the desired quantile value $\tau$.
1: Initialize $f^{[0]}(\cdot)$ with $f^{[0]}(\cdot) = 0$.
2: for $m = 1$ to $M$ do
3:     Compute the gradient $\frac{\partial}{\partial f} l(Y, f)$ and evaluate at $f^{[m-1]}(\mathbf{x}_i)$:
           $U_i = \frac{Y_i - (1-\tau)}{h} K'\left(\frac{f^{[m-1]}(\mathbf{x}_i)}{h}\right)$
4:     Fit the gradient vector $U_1, \cdots, U_n$ to $\mathbf{x}_1, \cdots, \mathbf{x}_n$ by the base procedure:
           $\{(\mathbf{x}_i, U_i),\ i = 1, \cdots, n\} \longrightarrow g^{[m]}(\cdot)$.
5:     Update the estimation by $f^{[m]}(\cdot) = f^{[m-1]}(\cdot) + \eta g^{[m]}(\cdot)$, where $\eta$ is a step-length factor.
6: end for
7: Output the obtained classifier $I(f^{[M]}(\mathbf{x}) \geq 0)$.
```

4.3 Related Works

Kordas et al. [13,14] proposed binary quantile regression to predict quantiles for classification tasks. However, in binary QReg, simulated annealing algorithm is employed to perform the optimization task. Although a local optimum is guaranteed, simulated annealing is well known for its expensive computation. While QBC is based on gradient ascent, which yields a local maximum and converges fast. Due to the expensive computation of simulated annealing, binary QReg can only work in very low dimensional spaces. However, in applications, we frequently face hundreds of, even thousands of predictors, and it is often desired to find out the informative predictors. Clearly, in this case, binary QReg is not applicable. On the contrary, QBC is designed to work in high dimensional spaces, and it enables us to select the most informative variables by using certain types of base learner.

Hall et al. [10] proposed a median based classifier which works in high dimensional space. For a given predictor vector, Hall et al. compares the L_1 distances from the new predictor vector to the component-wise medians of the positive and negative examples in the training set, and assigns class label as the class with the smaller L_1 distance. Although computationally efficient, this simple nearest neighbor like algorithm cannot perform variable selection as the proposed QBC algorithm. The method proposed in [15] can also be applied to classification tasks, however, it is computationally more expensive than QBC, and it is not clear how to select most informative predictors as well.

5 Experiments

This section tests the proposed QBoost algorithms on various regression and classification problems. In the generic gradient descent/ascent algorithm, the step size factor is of minor importance as long as it is small [1]. Thus, in all the following experiments, the step size parameter of QBR and QBC is fixed at $\eta = 0.1$.

5.1 Results of QBR

QBR was tested on five datasets from UCI machine learning repository: White Wine Quality (size: 4898, dimension: 11), Red Wine Quality (size: 1599, dimension: 11), Forest Fire (size: 517, dimension: 12), Concrete Slump (size: 103, dimension: 10), and Concrete Compressive (size: 1030, dimension: 9). The predictor variables and the response variables were normalized to be in $[-1, 1]$. The original QReg was also tested on these datasets. To make the comparison between QBR and QReg fair, we used simple linear regression as base procedure in QBR.

Table 1. The comparison between QReg and QBR on the 5 datasets from UCI machine learning repository. The listed are the mean values of the check losses of the 500 runs, and the standard deviations are listed in parentheses.

Dataset	$\tau = 0.25$		$\tau = 0.50$		$\tau = 0.75$	
	QReg	QBR	QReg	QBR	QReg	QBR
Red Wine	19.71(1.11)	19.41(1.11)	24.30(1.06)	24.07(1.09)	19.40(0.81)	19.16(0.79)
White Wine	62.40(1.84)	62.23(1.91)	78.69(1.78)	78.60(1.92)	62.03(1.50)	61.78(1.64)
Forest Fire	4.97(0.54)	4.93(0.54)	10.04(0.82)	9.62(0.78)	9.56(0.92)	9.14(0.73)
concrete slump	0.74(0.14)	0.71(0.12)	1.00(0.17)	0.92(0.16)	0.95(0.19)	0.84(0.17)
concrete comp.	15.02(0.77)	14.91(0.79)	18.21(1.08)	18.17(1.00)	13.63(0.83)	13.52(0.77)

To numerically evaluate the performances, in lack of the true quantile functions for these datasets, we adopt the sum of the "check loss" on the testing set as the error measure, which is defined as

$$L(\tau) = \sum_{i=1}^{N_{\text{test}}} \rho_\tau(Y_i - \hat{f}_\tau(\mathbf{x}_i)),$$

where N_{test} is the size of the testing set, and $\hat{f}_\tau(\mathbf{x}_i)$ is the predicted τ-th quantile at $\mathbf{x}_i$. By the definition of quantile, the smaller the value of $L(\tau)$ is, the closer the estimated quantile to the true value of quantile.

For each dataset, we randomly select 80% of the examples as training set, and the remaining 20% as testing set. QBR and QReg are separately trained and evaluated on these subsets. The partition-training-testing process is repeated 500 times. The means and the standard deviations of the 500 check losses are calculated, as shown in Table 1. It is readily observed that in all experiments, QBR uniformly achieves smaller average check loss compared to QReg, which indicates that QBR estimates more accurately.

In our experiments, both QReg and QBR obtain a linear function of the predictors. Since the predictors and response are normalized to be in $[-1, 1]$, it makes sense if we delete the variables with too small coefficients, thus performing variable selection. 20 noisy predictors are added to the Concrete Slump data, each is generated uniformly at random from $[-1, 1]$. These noisy predictors are considered as noninformative, and the original predictors are informative to the problem. Then we repeat the above experiment. In the obtained models, we calculate the sum of the absolute values of all the coefficients. For any predictor, if its coefficient absolute value is less than 1% of that sum, it is trimmed out. We calculate the average numbers of the selected noisy predictors of QReg and QBR over the 500 runs, the means and standard deviations of the check losses on testing set are also calculated. Table 2 summarizes the results, from which it is readily observed that for each quantile value, compared to QReg, QBR selects far fewer noisy predictors while achieves smaller mean error. This experiment shows that QBR is more robust to noisy predictors and keeps the variable selection ability of boosting algorithm.

Table 2. Comparison between QReg and QBR on variable selection: see text for details

Method	$\tau = 0.25$		$\tau = 0.50$		$\tau = 0.75$	
	# of N. V.	Error	# of N. V.	Error	# of N. V.	Error
QReg	8.13	0.99 (0.16)	6.97	1.38 (0.21)	9.69	1.28 (0.24)
QBR	1.04	0.97 (0.19)	0.63	1.14 (0.21)	0.98	1.27 (0.30)

The procedure of QBR enables us to select other forms of base learner, for example, regression trees [8,9], and this provides us flexibility in certain applications. While it is not clear how to make use of other forms of regression in the framework of the original QReg.

5.2 Results of QBC

In our experiments for classification, we made decision at the cut-off posterior probability 0.5, therefore we fit QBC with $\tau = 0.5$. The standard normal cumulative distribution function in Eqn. (11) was used as approximation to the indicator function with $h = 0.1$. From our experience, QBC is not sensitive to the value of h as long as $h < 0.5$. For the comparative purpose, since QBC is a boosting based procedure, we mainly considered several popular boosting based classifiers which include L_2-Boost [2], AdaBoost [7], and LogitBoost [9]. We also tested the Median Classifier [10] and compared the performance.

Results on Credit Score Data. We compare the result of QBC to that of binary QReg [13] on the German bank credit score dataset which is available from UCI machine learning repository. The dataset is of size 1000 with 300 positive examples, each example has 20 predictors normalized to be in $[-1, 1]$. In [13], only 8 predictors are preselected to fit the binary QReg due to the expensive simulated annealing algorithm it employees in the optimization procedure.

To run QBC, we randomly split the whole dataset without variable preselection into training set of size 800, and evaluate the learned QBC classifier on the other 200 examples. The QBC training algorithm was ran for 100 iterations using simple linear regression with only one predictor as base learner, by this way, it is fair to compare the performance of QBC to that of binary QReg. The splitting-training-testing process was repeated 500 times, and we report the mean training and testing error rates in Table 3, which also lists the performance of binary QReg from [13]. We see that QBC performs better than binary QReg on both the training and testing set. This is due to the efficient computation of QBC which allows the algorithm to explore more predictors and thus selects more informative ones.

In order to test the variable selection ability of QBC, 20 noisy predictors generated from uniform distribution on $[-1, 1]$ are added to the original dataset,

Table 3. Comparison between binary QReg and QBC on credit dataset. Clean means the original dataset, and Noisy means 20 noisy predictors are added to the dataset.

Dataset	QBC		Binary QReg	
	Training Error	Testing Error	Training Error	Testing Error
Clean	19.84%	24.47%	21.9%	26.5%
Noisy	22.53%	25.64%	NA	NA

Table 4. The testing errors of L_2 Boost, AdaBoost, LogitBoost, QBC, and Median Classifier on the credit dataset

Data	L_2 Boost	AdaBoost	LogitBoost	QBC	Median Classifier
Clean	28.68%	29.99%	28.81%	**28.50%**	35.04%
Noisy	31.92%	30.05%	32.68%	**28.55%**	38.94%

Table 5. The 5-fold cross validation mean error rates of the considered algorithms on the three bioinformatics datasets. All the algorithms were ran for 100 iterations. The best mean error rates for a dataset are displayed in **bold**.

Dataset	Size	dim	L_2-Boost	AdaBoost	LogitBoost	QBC	Median Classifier
Estrogen	49	7,129	21.11%	17.11%	15.11%	**13.33%**	13.38%
Colon	62	2,000	24.62%	21.41%	21.41%	21.67%	**14.58%**
Nodal	49	7,129	31.78%	24.89%	20.44%	**20.00%**	42.84%

and the above procedure was repeated 500 times. Table 3 also lists the average training and testing errors of QBC with only 1 noisy variable selected in average. Our result demonstrates that QBC performs only slightly worse on noisy data, which indicates that QBC is robust to noise. Due to the expensive computation of binary QReg, its performance on noisy data is not provided.

Compared to binary QReg, QBC enjoys the flexibility to choose other forms of weak learner. We also compared QBC to the alternatives mentioned at the beginning of Section 5.2 on the credit dataset with and without noisy predictors. All the boosting based algorithms used stump as base procedure for fair comparison. All algorithms were ran for 500 times with randomly selected 800 training examples and 200 testing examples, and the average testing error rates are listed in Table 4, from which we can see that compared to the alternative methods, QBC achieves the best performance on both clean and noisy data. Again, we observe that QBC deteriorates only slightly on the noisy data, which verifies its robustness to noisy predictors and is able to select informative variables.

Results on Bioinformatics Data. We compare QBC to the alternative methods on 3 publicly available datasets in bioinformatics [3]: Estrogen, Nodal, and Colon. All of the datasets are of very high dimension (see Table 5), and this makes binary QReg [13,14] not affordable. All the boosting-based algorithms used decision stump as base learner for fair comparison.

We have conducted 5-fold cross validation (5-CV), and Table 5 lists the average testing error rates for the 5 runs of each algorithm on every dataset. We observe that QBC yields the best performance on 2 out of the 3 datasets. On the Colon dataset, QBC performs better than L_2 Boost, similar to LogitBoost and AdaBoost, and worse than Median Classifier. It can also be observed that the Median Classifier [10] is not stable – sometimes the performance is very good, sometimes the performance is very poor.

6 Conclusion and Future Works

The original quantile regression (QReg) is inefficient when the predictor is in high dimensional space. This paper applies functional gradient descent to fit QReg, resulting Quantile Boost Regression (QBR). In the case of binary classification, the class label is defined via a hidden variable, and predicting the quantiles of the binary response is reduced to predicting the corresponding quantiles of the hidden variable. An equivalent form of the definition of quantile is introduced, whose smoothed version is employed as the objective function. Functional gradient ascent is used to maximize the objective function for fitting the classifier, yielding Quantile Boost Classification (QBC) algorithm. In QBR (QBC), the gradient of the objective function is approximated by a base procedure and the model grows along the negative gradient (gradient) direction. The proposed algorithms yield a local optimum, and converge fast, more importantly, they enable us to solve problems in high dimensional spaces.

Extensive experiments were conducted for regression and binary classification. Detailed analysis of the results show that QBR/QBC performs better than the original/binary QReg. Moreover, QBR and QBC are more robust to noisy predictors and able to select informative variables. On high dimensional datasets in bioinformatics, binary QReg is not applicable due to its expensive computation, while QBC performs better when compared to other alternatives.

QBR belongs to stage-wise additive model, which has a close connection to L_1 constrained models [6]. Recently, L_1 quantile regression models [16,17] were proposed which imposes L_1 constraint on the coefficient vector in quantile regression. Therefore, it is natural to investigate the relation between QBR and L_1 QReg in the aspects of performance and variable selection ability. The current version of QBC is for two-class problems, and we plan to develop the multi-class version of QBC by reducing it to a series of two-class tasks, for example, by one-vs-all [7] or Error-Correcting Output Codes [4].

Acknowledgement

This work was supported by CNAS Summer Research Fellowship of Missouri State University.

References

1. Bühlmann, P., Hothorn, T.: Boosting Algorithms: Regularization, Prediction and Model Fitting. Statistical Science 22, 477–505 (2007)
2. Bühlmann, P., Yu, B.: Boosting with the L2 Loss: Regression and Classification. Journal of the American Statistical Association 98, 324–340 (2003)
3. Dettling, M., Bühlmann, P.: Boosting for Tumor Classification with Gene Expression Data. Bioinformatics 19(9), 1061–1069 (2003)
4. Dietterich, T.G., Bakiri, G.: Solving Multiclass Learning Problems via Error-Correcting Output Codes. Journal of Artificial Intelligence Research 2, 263–286 (1995)
5. Duffy, N., Helmbold, D.: Boosting Methods for Regression. Machine Learning 47(2-3), 153–200 (2002)
6. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least Angle Regression. Annals of Statistics 32(2), 407–499 (2004)
7. Freund, Y., Schapire, R.: A Decision-Theoretic Generalization of On-line Learning and An Application to Boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)

8. Friedman, J.: Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics 29, 1189–1232 (2001)
9. Friedman, J., Hastie, T., Tibshirani, R.: Additive Logistic Regression: a Statistical View of Boosting. Annal of Statistics 28(2), 337–407 (2000)
10. Hall, P., Titterington, D.M., Xue, J.H.: Median-based Classifiers for High-Dimensional Data (2008), http://www.gla.ac.uk/media/media_71170_en.pdf
11. Hunter, D.R., Lange, K.: Quantile Regression via an MM Algorithm. Journal of Computational & Graphical Statistics (2000)
12. Koenker, R.: Quantile Regression. Cambridge University Press, Cambridge (2005)
13. Kordas, G.: Credit Scoring Using Binary Quantile Regression. In: Statistical Data Analysis Based on the L_1-Norm and Related Methods (2002)
14. Kordas, G.: Smoothed Binary Regression Quantiles. Journal of Applied Econometrics 21(3), 387–407 (2006)
15. Langford, J., Oliveira, R., Zadrozny, B.: Predicting Conditional Quantiles via Reduction to Classification. In: In Proc. of Uncertainty in Artificical Intelligence (2006)
16. Li, Y., Zhu, J.: L_1-Norm Quantile Regression. Journal of Computational & Graphical Statistics 17(1), 163–185 (2008)
17. Wu, Y., Liu, Y.: Variable Selection in Quantile Regression. Statistica Sinica 19, 801–817 (2009)
18. Zemel, R., Pitassi, T.: A Gradient-based Boosting Algorithm for Regression Problems. In: Proc. of Advances in Neural Information Processing Systems (2001)

A Fast Fuzzy Cocke-Younger-Kasami Algorithm for DNA and RNA Strings Analysis

Herón Molina-Lozano

Centro de Investigación en Computación, Instituto Politécnico Nacional, Av. Juan de Dios Bátiz s/n casi esq. Miguel Othón de Mendizábal. Unidad Profesional Adolfo López Mateos Col. Nueva Industrial Vallejo Delegación Gustavo A. Madero C.P 07738, México D.F.
hmolina@ipn.mx

Abstract. In this paper we present a variation of the Cocke-Younger-Kasami algorithm (CYK algorithm for short) for the analysis of fuzzy free context languages applied to DNA strings. We propose a variation of the original CYK algorithm where we prove that the order of the new CYK algorithm is O(n). We prove that the new algorithm only use $2n$-1 memory localities. We use a variation of the CYK algorithm, where the free context language can be fuzzy. The fuzzy context-free grammar (FCFG) is obtained from DNA and RNA sequences.

Keywords: Pattern recognition, Cocke-Younger-Kasami algorithm, fuzzy logic, ADN, RNA.

1 Introduction

Context-free grammars are of great importance in formalizing the notion of parsing, simplifying translation of programming languages, and other string-processing applications [5]. Context-free grammars have been used to generate sequences of characters in a string. Recognition using grammars is formally very similar to the general approaches used throughout pattern recognition. Is possible, to detect if a string belongs to a specific CFG by using the CYK algorithm [5] and [6]. The CYK algorithm is a bottom-up parsing method. This method starts with a test sequence and seeks to simplify it, so as to represent it with a root (starting) symbol. The CYK algorithm is a method used to fills a parse table (a matrix) from the bottom-up. The grammar must be expressed in Chomsky normal form. After the CYK algorithm has been applied the table must contain the source symbol. The existence of the source symbol in the table indicates that the string belongs to the language of the free-grammar, and the string is accepted, on the contrary, the string is rejected and does not belong to the grammar. It was a natural step to use the theory of grammar to represents different kinds of biological strings, from DNA, RNA to proteins. D. Serls shows different methodologies used by the Linguistic Sciences in order to analyze strings by using crisp grammars [7]. Specifically, by using context free grammars is possible to generate languages that describe DNA strings. The CFGs have been used to compare DNA substrings [8] and [9], describe folded RNA secondary structures [10] and [11], specification of gene regulatory elements [12], among other applications. The same DNA sequence from two different living organisms of the same species could be different, this situation is possible because there are some evolutive differences that are produced by mutations that occurred over the course of time, but essential similarities are maintained and preserved. From the perspective of the theory of fuzzy logic these DNA sequences are elements that could belong to a CFG but with different grade of membership. The theory of fuzzy logic gives the possibility to handle imprecision and supported with the theory of grammars, could be applied to recognize molecular biology sequences based on the use of the FCYK algorithm.

The algorithm finds the membership of a string. When the string belongs to a context-free grammar (CFG), the string has a membership of 1, but when the string does not belong to the context-free language (CFL) the string has a membership of 0. There is a possibility to handle not only memberships that have a value of 1 or 0, but also to handle memberships that are among 0 and 1. E. Lee and L. Zadeh proposed the use of fuzzy logic in order to handle CFG that have the fuzzy attribute [1]. In that sense, is possible to determine a fuzzy context-free language (FCFL) that can give to a string a membership near of 1, the string is a little different from the fuzzy grammar, or a membership near 0, the string is very different from the fuzzy grammar. P. Asveld proposed to use the CKY algorithm to recognize FCFL [2]. We proposed to use the fuzzy CKY (FCKY) algorithm to recognize DNA sequences (strings of nucleotides) [3]. This paper is an extension of [3]. The number of nucleotides of a DNA could be some units, hundreds, thousands, or even millions, depending on the kind of DNA sequence. The use the CKY algorithm could be impractical when the number of nucleotides is hundred or thousands of bases, because the order of the algorithm is O(n^3), where n is the number of symbols of the string [4], in the context of this paper the symbols are DNA nucleotides. Also, the memory space of the CYK algorithm is n^2. In section III we prove that for the analysis of DNA sequences with the FCKY algorithm the order is O(n^3/6), starting from this point of view, we propose to modify the FCKY algorithm in order to analyze DNA sequences, the order of the modified FCKY is reduced to O(n), and the memory space is $2n$-1, thus the modified algorithm can do fast DNA sequences analysis. Now we are going to talk about some antecedents that support the paper.

In the next sections we give the basic theory of CFG, FCFG, the CYK algorithm and the FCYK algorithm. Next we prove that the order of FCYK applied to DNA sequences is O(n^3/6). We propose a modification to the FCYK algorithm where the order is O(n^2) and the number of memory locations is n^2. Also, we propose a modification to the FCYK algorithm where the order is O(n^2) and the number of memory locations is $2n$-1. An example applied to DNA sequences is presented. Finally, the conclusions are presented.

2 Fuzzy Context Free Grammars

2.1 Type of Grammars

In general a (crisp) grammar is defined by an alphabet, a set of variables, the set of production rules, and the starting symbol (4-tuple): $G = (N, \Sigma, P, S)$, where N is the finite set whose elements are called nonterminal symbols, Σ is a finite set whose elements are called terminal symbols (where $N \cap \Sigma = 0$), P is the finite subset of $[(N \cup \Sigma)^{*} \backslash \Sigma^{*}] \times (N \cup \Sigma)^{*}$, called the set of productions, and $S \in N$ is called the starting symbol. Grammars are classified according to the form of the productions rules used (Chomsky hierarchy) [13]. The language generated by the grammar denoted by $L(G)$, is the set of all the possible strings (maybe an infinite set) that can be generated by G. There are four principal type of grammars, and are classified according to the form of the productions rules. A production rule is of the form $\alpha \rightarrow \beta$, where α and β are string of characters, where the former are nonterminals symbols, and the latter are terminal symbols [6].

A language generated by a grammar of type i is called a type i. It can be shown that the class of grammars of type i includes all grammars of type $i + 1$; thus there is a strict hierarchy in grammars.

Once that the CFG has been defined is possible to define the grade of membership $L(G)$ of a DNA string $\boldsymbol{x}$. In this case a grammar G generated from a DNA sequence has the next elements: N is a set of nonterminal symbols, some of these symbols are A, C, G and T, where these symbols represents the nucleotides adenine, cytosine, guanine and thymine. Σ is the set of terminal symbols, these symbols are a, c, g, t. P are the productions, and S is the starting symbol. In order to prove if a string $\boldsymbol{x}$ belongs (has a grade of membership of 1) to a grammar G, the CYK algorithm has been proposed. In the case that the string $\boldsymbol{x}$ don't belong to the grammar G, the string $\boldsymbol{x}$ has a grade of membership equal to 0. We must consider that the kind of grammar that the CYK algorithm uses is of type 2 (context-free) in Chomsky normal form (CNM). In the next section we define the CNM.

2.2 Chomsky Normal Form

Theorem 1. (CNF) Any free context language $L(G)$, without the empty word λ, can be generated from the grammar G, where the production rules have the next characteristics: $A \rightarrow BC$ or $A \rightarrow a$. Where A, B, C are variables, and a is a terminal symbol ($a \in \Sigma$). This theorem is proved in [5] and [13].

Example 1. Consider a DNA sequence $\boldsymbol{x}$ = "agctacg". We need to generate a grammar G_1 in CNF. Is necessary to define the 4 the terminal symbols: a, c, g and t. Also, we need to define the non terminal symbols: A, C, G and T. Also we add the initial symbol S. Formally, the grammar is as follows: $G_1 = (\{S, A, C, G, T, W_1, W_2, W_3, W_4, W_5, a, c, g, t\}, \{a, c, g, t\}, P_1, S)$, where P_1 are the next production rules:

$S \rightarrow AW_1$
$W_1 \rightarrow GW_2$
$W_2 \rightarrow CW_3$
$W_3 \rightarrow TW_4$
$W_4 \rightarrow AW_5$
$W_5 \rightarrow CG$
$A \rightarrow a$
$C \rightarrow c$
$G \rightarrow g$
$T \rightarrow t$

Now we prove that starting from S and using the productions rules of G_1 we obtain $L(G_1)$ = "agctacg":

2.3 Fuzzy Context-Free Grammars

Informally, a fuzzy grammar may be viewed as a set of rules for generating the elements of a fuzzy subset [13]. A fuzzy grammar is a 4-tuple $G = (N, \Sigma, P, S)$, where as same as a crisp CFG N is a set of nonterminal symbols, Σ is the set of terminal symbols ($N \cap \Sigma = 0$), P is the set of fuzzy productions, and S is the starting symbol ($S \in N$). The productions here are of the form $I \to x/m$, formally $\mu(I \to x) = m$, where $m > 0$ is the grade of membership of x given I , $I \in N$, $\mathrm{x} \in (N \cup \Sigma)^*$ (other that the empty word λ), and $S \to \lambda$. Essentially, the elements of N are labels for certain fuzzy subsets of Σ^* called fuzzy syntactic categories, with S being the label for the syntactic category "sentence". The elements of P define conditioned fuzzy subsets of $(N \cup \Sigma)^*$. The expression $I \to x/m$ represents a rewrite rule, where x is directly derivable from I with a grade of membership m. If $x_1, \ldots, x_2$ are strings in $(N \cup \Sigma)^*$ and:

$$x_1 \to x_2/m_2, x_{c-1} \to x_c/m_{c,} m_2, \ldots, m_c > 0,$$

then x_1 is said to derive x_c in grammar G, or equivalently, x_m is derivable for x_1 in grammar G. The expression:

$$x_1 \to x_2/m_2 \to \ldots \to \mathrm{x}_{c-1}/m_{c-1} \to x_c/m_c$$

is referred to as a derivation chain from x_1 to x_c.

A fuzzy grammar G generates a fuzzy language $L(G)$ in the following manner. A string of terminals x is said to be in $L(G)$ if and only if x is derivable from S. The grade of membership of x in $L(G)$ is given by:

$$\mu_G(x) = \vee\,(\mu\,(S, x_1) \wedge \mu\,(x_1, x_2) \wedge \ldots \wedge \mu(x_c, x)) \qquad (1)$$

where the union operator $\vee$ (*supremum*) is taken over all derivation chains from S to x. Thus (1) defines $L(G)$ as a fuzzy subset of $(N \cup \Sigma)^*$. Equation (1) may be expressed as follows: a) $\mu_G(x)$ is the grade of membership of x in the language generated by grammar G; b) $\mu_G(x)$ is the strongest derivation chain from S to x. The operator $\vee$ in reality represents a T-conorm operator [14].

3 Fuzzy Cocke-Younger-Kasami Algorithm

3.1 Cocke-Younger-Kasami Algorithm

In this section we discuss the functional version of the CYK algorithm for recognizing (crisp, ordinary, non-fuzzy) CFL [2]. Usually, the CKY algorithm is presented in terms of nested "**for**-loops" filling an upper-triangular matrix (table). Given a CFG $G = (N, \Sigma, P, S)$ in CNF (without the empty word λ) and a string $a_1a_2 \ldots a_n$ ($n \geq 1$) with $a_k \in \Sigma (1 \leq k \leq n)$. Fill the strictly upper-triangular $(n + 1) \times (n + 1)$ recognition matrix $\boldsymbol{T}$ by the algorithm, where each element $\mathrm{t}_{\mathrm{i,j}}$ is a subset of V if $V = N - \Sigma$ and is initially empty.

Algorithm 1. CKY

```
1. for $i \leftarrow 0$ to $n$-1
2.     $t_{i,\, i+1} \leftarrow \{A \mid a_{i+1} \in P(a)\}$;
3. for d $\leftarrow 2$ to $n$
4.     for $i \leftarrow 0$ to $n$-$d$
5.         $j \leftarrow d + i$;
6. $t_{i,j} \leftarrow t_{i,j} \cup \{A \mid \exists k\ (i+1 \le k \le j-1); \exists B(B \in t_{i,\, k}); \exists C(C \in t_{k,\, j}); BC \in P(A)\}$;
7. return ($t_{0,\, n}$)
```

Then the string $a_1a_2 \ldots a_n \in$L(G) if and only if S $\in$ $t_{0,n}$.

Example 2. Consider the sequence $\boldsymbol{x}$ = “aggt”, over Σ, and the GFC G_2 = ({S, A, C, G, T, W_1, W_2, a, c, g, t}, { a, c, g, t }, P_2, S), with P_2 in CNF:

$P_2(S) = \{S, AW_1\}$
$P_2(W_1) = \{W_1, GW_2, TW_2\}$
$P_2(W_2) = \{W_2, GT, AC\}$
$P_2(A) = \{A, a\}$
$P_2(C) = \{C, c\}$
$P_2(G) = \{G, g\}$
$P_2(T) = \{T, t\}$

We can observe at table I the initial step of the algorithm (steps 1-2 from the algorithm) that $t_{0,\,1} = A$, $t_{1,\,2} = G$, $t_{2,\,3} = G$ y $t_{3,\,4} = T$. Then applying the second step of the algorithm (steps 3-6) the string $\boldsymbol{x}$ = “aggt” is recognized due that $S \in t_{0,4}$.

Table 1. String x = “AGGT” recognized with algorithm 1

i/j	1	2	3	4
0	A	$\varnothing$	$\varnothing$	S
1		G	$\varnothing$	W_1
2			G	W_2
3				$T/1$

Example 3. Now we going to consider that the string $\boldsymbol{x}$ is “acgt”. Then $t_{0,\,1} = A$, $t_{1,\,2} = C$, $t_{2,\,3} = G$ y $t_{3,\,4} = T$. Table II shows that $\boldsymbol{x}$ $= acgt \notin L(G_2)$, because $S \notin t_{0,\,4}$.

Table 2. String x = “ACGT” not recognized

i/j	1	2	3	4
0	A	$\varnothing$	$\varnothing$	$\varnothing$
1		C	$\varnothing$	$\varnothing$
2			G	W_2
3				T

Once we have the CYK algorithm in a functional form (algorithm 1), it is easy to obtain a robust modification for recognizing FCFL.

3.2 The CYK Algorithm for FCFL

Given a CFG $G = (N, \Sigma, P, S)$ in CNF (without the empty word λ) and a string $a_1a_2 \ldots a_n$ $(n \geq 1)$ with $a_k \in \Sigma$ $(1 \leq k \leq n)$. Fill the strictly upper-triangular $(n + 1) \times (n + 1)$ recognition matrix $\boldsymbol{T}$ by the algorithm, where each element $t_{i,j}$ is a subset of V if $V = N - \Sigma$ and is initially empty.

Algorithm 2. FCKY

```
1.  for $i \leftarrow 0$ to $n$-1
2.          $t_{i,\, i+1} \leftarrow \{(A, m) \mid A \in N, \mu(a_{i+1}; P(a)) = m > 0\}$;
3.  for d $\leftarrow$ 2 to $n$
4.          for $i \leftarrow 0$ to $n$-$d$
5.                  $j \leftarrow d + i$;
6.  $t_{i,j} \leftarrow t_{i,\,j} \cup \{(A, m) \mid A\in N, m = \vee \{ r * p * q \mid (B, p) \in t_{i,\,k}; (C, q) \in t_{k,\,j}; \mu(BC,$
    $P(A)) = r > 0; i{+}1 \leq k \leq j{-}1\}\}$;
7.  return $(t_{0,\,n})$
```

Then for each $m > 0$ $(m \in L)$, $\mu(a_1a_2\ldots a_n; L(G)) = m$, if and only if $(S, m) \in t_{0,\,n}$. From the algorithm 2, m is the grade of membership of a rule P. The operation $*$ represents a T-norm, and could be substituted by any of the four T-norms. In our case we use the Tmin, and Tap T-norm operators.

Example 3. Consider a fuzzy context free grammar $G_3 = (V_1, \Sigma_1, P_3, S)$ with $\Sigma_1 = (a, c, g, t)$, $V_1 = \Sigma_1 \cup \{S, A, C, G, T, W_1, W_2\}$, where P_3 has the next set of rules:

$S \rightarrow SS \mid AW_1 \mid CW_1$
$W_1 \rightarrow GW_2 \mid TW_2 \mid AW_2$
$W_2 \rightarrow AT \mid CG$
$A \rightarrow a$
$C \rightarrow c$
$G \rightarrow g$
$T \rightarrow t$

Where $\mu(CW_1, P_1(S)) = 0.9$, $\mu(TW_2, P_1(W_1)) = 0.8$, $\mu(AW_2, P_1(W_1)) = 0.1$, $\mu(CG, P_1(W_2)) = 0$. and the remainder rules have a μ value equal to 1. If we have the string $\boldsymbol{x}$ = "ctat" over Σ_1. We see that the string "ctat" has two grammatical errors ("tiny errors"). In order to star the algorithm 1 we initialize with $t_{0,\,1} = \{(C/1)\}$, $t_{1,\,2} = \{(T/1)\}$, $t_{2,\,3} = \{(A/1)\}$ y $t_{3,\,4} = \{(T/1)\}$, and using the iterative step we obtain the Table III with the next results: Due to $(S/0.72) \in t_{0,\,4}$ the string have a membership value of $\mu(ctat; L(G_3)) = 0.72$.

Table 3. String x = "CTAT" recognized with algorithm 2

i/*j*	1	2	3	4
0	$C/1$	∅	∅	$S/0.72$
1		$T/1$	∅	$W_1/0.8$
2			$A/1$	$W_2/1$
3				$T/1$

Now that we have used the FCYK algorithm to give the grade of membership to a DNA sequence, is possible to calculate the number of iterations that are used by this kind of strings.

3.3 Computational Order of the FCYK Algorithm Applied to DNA Sequences

M. A. Harrison has calculated the order of the original CYK algorithm [14], and in general is equal to O(n^3). Even though, theorem shows that the algorithm 2 at the moment to use DNA sequences the computational order is reduced.

Proposition. For a CFG that is obtained from a DNA sequence x = $\boldsymbol{x} = a_1a_2\ldots a_n$ in CNF, the productions rules obtained have the form:

$$\begin{aligned}&\mathrm{S} \rightarrow \mathrm{A_iW_k}\\&\mathrm{W_k} \rightarrow \mathrm{A_iW_l}\\&\mathrm{W_l} \rightarrow \mathrm{A_iA_j}\\&\mathrm{A_i} \rightarrow \mathrm{a_i}\\&\mathrm{A_j} \rightarrow \mathrm{a_j}\end{aligned}$$

where A_i, A_j, W_k, W_l are nonterminal symbols, S is the starting symbol, a_i, a_j are terminal symbols, and n is the number of nucleotides of the sequence. Then the order of the CYK algorithm is equal to:

$$\frac{n}{6}\left(n^2+5\right) \tag{2}$$

Proof. For a DNA sequence $\boldsymbol{x}$ with n nucleotides the CYK algorithm do the first instruction **for** in n steps. Next the CYK algorithm has two instructions **for**, so the next instructions are added (n-1)(n-(n-1)) + (n-2)(n-(n-2)) + … + (n-(n-1))(n-1), in general the total number of instructions realized by the CYK algorithm is calculated as:

$$\begin{aligned}&n+\sum_{i=1}^{n-1}(n-i)i = n+\sum_{i=1}^{n-1}\left(ni-i^2\right)=\\&n+n\sum_{i=1}^{n-1}i-\sum_{j=1}^{n-1}j^2\end{aligned} \tag{3}$$

We know that:

$$\sum_{i=1}^{n}i=\frac{n(n+1)}{2} \quad \text{and} \quad \sum_{i=1}^{n}i^2=\frac{n(n+1)(2n+1)}{6}$$

Therefore eq. (3) is equal to:

$$n+n\left[\frac{(n-1)(n-1+1)}{2}\right]-\left[\frac{(n-1)n(2(n-1)+1)}{6}\right]=$$
$$n+\frac{n^3-n^2}{2}-\frac{2n^3-3n^2+n}{6}=\frac{n^3}{6}-\frac{5n}{6} \tag{4}$$

Q.E.D.

Corollary 1. For all $n \in N$, $\frac{n}{6}\left(n^2+5\right)<n^3$, with $n \geq 1$.

Proof. We have that:

$$n\left(n^2+5\right)<6n^3 \tag{5}$$

Where:

$$n^3+5n-6n^3<0 \tag{6}$$

Making some algebra:

$$-5n^3+5n<0 \tag{7}$$

Because $n \in N$, where n ≥ 1, therefore:

$$\lim_{n\to\infty} 5n\left(-n^2+1\right)<0 \tag{8}$$

Where $5n$:

$$\lim_{n\to\infty} -n^2+1<0 \tag{9}$$

Then we have that -∞ < 0.

Q.E.D.

From the corollary 1 we have that the number of instructions realized by the CYK algorithm in CNF using DNA sequences is less than n^3.

From these results we observed the possibility to modify the FCKY algorithm in order to reduce the order of this algorithm. In the next section we show the results.

4 Modified FCYK Algorithm

4.1 The First Version of the Fast FCYK Algorithm

We observed that the elements of the matrix ***T*** at the moment that the CYK algorithm is applied to recognize DNA sequences are the same. That means that first the diagonal of the upper-diagonal matrix is filled, then only the last column of the matrix is filled until the element $t_{0,n}$ is reached. The fig. 1 shows the pattern that is followed from the CYK nor FCYK algorithm don't matter the number n of elements of the sequence. The first arrows represent the set of all the elements $t_{i,i+1}$ of the CYK algorithm. The second arrow represents all the elements $t_{i,j}$ of the algorithm.

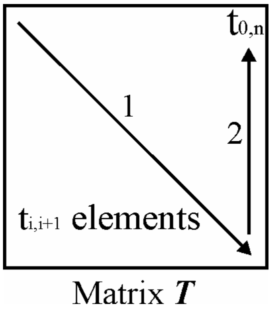


Fig. 1. Upper-diagonal matrix ***T*** of the CYK algorithm

The first algorithm modifies only the second group of the "**for**" instructions. One instruction "**for**" can be eliminated. Now the first version of the fast CKY algorithm is presented.

Given a CFG $G = (N, \Sigma, P, S)$ in CNF (without the empty word λ) and a string $a_1a_2 \ldots a_n$ ($n \geq 1$) with $a_k \in \Sigma$ ($1 \leq k \leq n$). Fill the strictly upper-triangular $(n + 1) \times (n + 1)$ recognition matrix ***T*** by the algorithm, where each element $t_{i,j}$ is a subset of V if $V = N - \Sigma$ and is initially empty.

Algorithm 3. Fast FCYK algorithm (version 1)

```
1.   for $i \leftarrow 0$ to $n$-1
2.   $t_{i,i+1} \leftarrow \{(A, m) \mid A \in N, \mu(a_{i+1}; P(a)) = m > 0\}$;
3.   for $l \leftarrow n$-2 to 0
4.   $t_{l,n} \leftarrow t_{l,n} \cup \{(A, m) \mid A \in N, m = \vee \{ r * p * q \mid (B, p) \in t_{l,l+1}; (C, q) \in t_{l+1,n};$
       $\mu(BC, P(A)) = r > 0\}\}$;
5.   return ($t_{0,n}$)
```

Then for each $m > 0$ ($m \in L$), $\mu(a_1a_2 \ldots a_n; L(G)) = m$, if and only if $(S, m) \in t_{0,n}$. As we can see, the first instruction **for** do n iterations, and the second instruction **for** do n-1 iterations, finally the number of iterations is $2n$-1. About the space in memory is necessary to use n^2 memory localities. From fig. 1 we observe that is possible to use only a vector of memory localities and not a matrix. Now we present the second modification of the FCYK algorithm.

4.2 The Second Version of the Fast FCYK Algorithm

From the fig.1 we can obtain a vector, fig. 2 shows that the vector is divided in two sections, in the first section are only the group of elements from t_0 to t_{n-1} (first instruction **for**) in the second group are the elements from t_n to t_{2n-2} (second instruction **for**). The symbol m_k ($1 \leq k \leq 2n$-1) represents the membership grade of each element. With that idea in mind is possible to reduce the number of memory localities to $2n$-1 as the fig. 2 indicates.

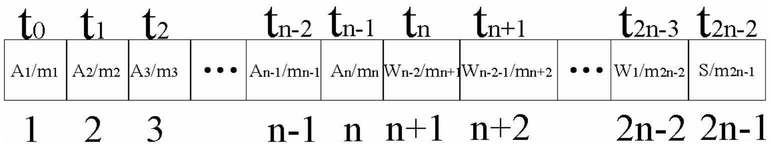

t0 t1 t2 tn-2 tn-1 tn tn+1 t2n-3 t2n-2
A1/m1 A2/m2 A3/m3 • • • An-1/mn-1 An/mn Wn-2/mn+1 Wn-2-1/mn+2 • • • W1/m2n-2 S/m2n-1
1 2 3 n-1 n n+1 n+2 2n-2 2n-1

Fig. 2. Vector ***t*** obtained from the second version of the fast FCYK algorithm

This is the second version of the fast FCYK algorithm.

Given a CFG $G = (N, \Sigma, P, S)$ in CNF (without the empty word λ) and a string $a_1a_2 \ldots a_n$ ($n \geq 1$) with $a_k \in \Sigma (1 \leq k \leq n)$. Fill the vector $\boldsymbol{t}$ $(2n - 1)$ by the algorithm, where each element t_l is a subset of V if $V = N - \Sigma$ and is initially empty.

Algorithm 4. Fast FCYK algorithm (version 2)

```
1.   for $i \leftarrow 0$ to $n$-1
2.        $t_i \leftarrow \{(A, m) \mid A \in N, \mu(a_{i+1}; P(a)) = m > 0\}$;
3. $r \leftarrow 2$
4.   for $l \leftarrow (n - 2)$ to 0
5.        $q \leftarrow n - r$
6.        $k \leftarrow l - 1$
7.        $r \leftarrow r + 1$
7.        $t_l \leftarrow \{(A, m) \mid A \in N, m = \vee \{ p * q * k \mid (B, q) \in t_q;$
          $(C, k) \in t_k; \mu(BC, P(A)) = p > 0\}\}$;
8.   return ($t_{2n-2}$)
```

Then for each $m > 0$ (where $m \in L$), $\mu(a_1a_2 \ldots a_n; L(G)) = m$, iff $(S, m) \in t_{2\,n-1}$. We can notice that instruction 7 there is not necessity for do the operation of union ($\cup$) with other elements of t. That means that for each vector locality there is only one element t.

Example 4. Consider the grammar $G_6 = (V_6, \Sigma_6, P_6, S)$, $V_6 = \{S, A, C, G, T, W_1, W_2, \ldots, W_5\}$ y $\Sigma_6 = \{a, c, g, t\}$ that represents the DNA sequence “gataca”. And the productions are:

$S \rightarrow GW_1 \mid AW_1/_{0.1} \mid CW_1/_{0.2} \mid TW_1/_{0.62}$
$W_1 \rightarrow AW_2 \mid CW_2/_{0.55} \mid GW_2/_{0.3} \mid TW_2/_{0.45}$
$W_2 \rightarrow TW_3 \mid AW_3/_{0.4} \mid CW_3/_{0.27} \mid GW_3/_{0.73}$
$W_3 \rightarrow AW_5 \mid CW_5/_{0.55} \mid GW_5/_{0.3} \mid TW_5/_{0.45}$
$W_4 \rightarrow CA \mid CC/_{0.55} \mid CG/_{0.3} \mid CT/_{0.45} \mid AA/_{0.51} \mid AC/_{0.51\times0.55} \mid AG/_{0.51\times0.3} \mid AT/_{0.51\times0.45} \mid GA/_{0.27} \mid GC/_{0.27\times0.55} \mid GG/_{0.27\times0.3} \mid GT/_{0.27\times0.45} \mid TA/_{0.25} \mid TC/_{0.25\times0.55} \mid TG/_{0.25\times0.3} \mid TT/_{0.25\times0.45}$
$A \rightarrow a$
$C \rightarrow c$
$G \rightarrow g$
$T \rightarrow t$

We want to know the grade of membership of the next string, $\boldsymbol{x}$ = "gctact". Applying the fast CYK algorithm (ver. 2) the results are showed on the Table 4.

Table 4. Sequence x = "GCTACT" recognized with algorithm 4

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}
G/	C/	T/	A/	C/	T/	W_4/	W_3/	W_2/	W_1/	S/
1	1	1	1	1	1	0.45	0.45	0.45	0.11	0.11

5 Conclusions

From the genetic point of view there is a possibility to found RNA strings. The RNA is a polymer with four different sub-units. The four nucleotides are A, C G and U, where U means uracilo. Is possible to obtain a fuzzy free-context grammar from RNA, but in this case the proposition for DNA string are no longer valid. But is possible that in a future work, to present different proposition for RNA strings.

We have showed the possibility to use the fuzzy Cocke-Younger-Kasami algorithm to recognize DNA sequences from a fuzzy free-context grammar. In the worst case, the number of iterations that the FCYK algorithm has with DNA sequences is $n/6(n^2 + 5)$.

Using the fast fuzzy version of the method proposed by Cocke-Younger-Kasami it is possible to determine the membership grade of a DNA string by using fuzzy context-free grammars with a computational order of $O(n)$, and a memory space of $2n$-1.

The fast FCYK algorithm can be used to recognize DNA sequences from a motif by using fuzzy grammars. Overall, we believe that fuzzy grammar can be useful for approaching other problems in bioinformatics and computational biology due to the inherent fuzziness and imprecision of biological systems at the molecular level. For example the FCYK algorithm could be used to compare substrings, to work with other bioinformatics inspired algorithm like LOGO and discover new DNA sequences and obtain the membership grade of RNA sequences in order to discover secondary structures.

Acknowledgement

This work was supported by the Instituto de Ciencia y Tecnología del Distrito Federal (ICyTDF) under award No. PICCT08-22. Any opinions, findings and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsoring agency.

References

1. Lee, E.T., Zadeh, L.A.: Note on Fuzzy Languages. Information Sciences 1, 421–434 (1969)
2. Asveld, P.R.J.: Fuzzy Context-Free Languages-Part 2: Recognition and Parsing Algorithms. Theoretical Computer Science 347, 191–213 (2005)

3. Molina-Lozano, H., Vallejo-Clemente, E., Morett-Sánchez, J.: DNA Sequence Analysis Using Fuzzy Grammars. IEEE World Congress on Computational Intelligence (2008)
4. Linz, P.: Formal Languages and Automata, 4th edn. Jones and Bartlett Publishers, USA (2006)
5. Hopcroft, J.E., Rajeev Motwai, R., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading (2002)
6. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley-Interscience, Hoboken (2001)
7. Searls, D.B.: The Languages of Genes. Nature 420, 211–217 (2002)
8. Brendel, V., Busse, H.: Genome Structure Described by Formal Languages. Nucleic Acids Research 12, 2561–2568 (1984)
9. Head, T.: Formal Languages Theory and DNA. Bulleting Mathematical Biology 49 (1987)
10. Searls, D.: The Linguistics of DNA. American Scientific 80, 579–591 (1992)
11. Searls, D.: Artificial Intelligence and Molecular Biology. In: Hunter, L. (ed.) pp. 47–120. AAAI Press, Menlo Park (1993)
12. Collado-Vides, J.: A Transformational Grammar Approach to the Study of the Regulation of Gene Expression. Journal of Theory Biology 136, 403–425 (1989)
13. Mordeson, J.N., Malik, D.S.: Fuzzy Automata and Languages: Theory and Applications. Chapman and Hall/CRC (2002)
14. Jang, J.-S., Sun, C.-T., Mitzutani, E.: Neuro-Fuzzy and Soft Computing: a Computational Approach to Learning and Machine Intelligence. Prentice-Hall, Englewood Cliffs (1997)
15. Jones, N.C., Pevzner, P.A.: An Introduction to Bioinformatics Algorithms. MIT Press, Cambridge (2004)
16. Database of Protein Domains, Families and Functional Sites, `http://www.expasy.ch/prosite`

A Fast Implementation of the CT_EXT Algorithm for the Testor Property Identification

Guillermo Sanchez-Diaz[1], Ivan Piza-Davila[2], Manuel Lazo-Cortes[3], Miguel Mora-Gonzalez[4], and Javier Salinas-Luna[1]

[1] Universidad de Guadalajara, Centro Universitario de los Valles, Carr. Guadalajara-Ameca, Km. 45, Ameca, Jal. Mexico, C.P. 46600
{guillermo.sanchez,javier.salinas}@profesores.valles.udg.mx

[2] Instituto Tecnologico y de Estudios Superiores de Occidente, Periferico Sur Manuel Gomez Morin 8585, Tlaquepaque, Jal. Mexico, C.P. 45604
hpiza@iteso.mx

[3] Universidad de las Ciencias Informaticas, Carr. de San Antonio de los Baños Km. 2.5, Torrens, Havana, Cuba
mlazo@cedai.com.cu

[4] Universidad de Guadalajara, Centro Universitario de los Lagos, Av. Enrique Diaz de Leon 1144, Lagos de Moreno, Jal. Mexico, C.P. 47460
mmora@culagos.udg.mx

Abstract. Typical testors are a useful tool for both feature selection and for determining feature relevance in supervised classication problems. Nowadays, generating all typical testors of a training matrix is computationally expensive; all reported algorithms have exponential complexity, depending mainly on the number of columns in the training matrix. For this reason, different approaches such as sequential and parallel algorithms, genetic algorithms and hardware implementations techniques have been developed. In this paper, we introduce a fast implementation of the algorithm CT_EXT (which is one of the fastest algorithms reported) based on an accumulative binary tuple, developed for generating all typical testors of a training matrix. The accumulative binary tuple implemented in the CT_EXT algorithm, is a useful way to simplifies the search of feature combinations which fulfill the testor property, because its implementation decreases the number of operations involved in the process of generating all typical testors. In addition, experimental results using the proposed fast implementation of the CT_EXT algorithm and the comparison with other state of the art algorithms that generated typical testors are presented.

Keywords: feature selection, typical testors, pattern recognition.

1 Introduction

Feature selection is a significant task in supervised classification and other pattern recognition areas. This task consists of identifying those features that provide relevant information for the classification process. In Logical Combinatorial

Pattern Recognition [6, 12], feature selection is solved using Testor Theory [4]. Yu. I. Zhuravlev introduced the utilization of the testor concept in pattern recognition problems [3]. He defined a testor as a set of features that does not confuse objects descriptions belonging to different classes. This concept has been extended and generalized in several ways [4]. This concept is especially well suited to problems which involve qualitative and quantitative features (mixed data) and even incomplete descriptions.

Computing all typical testors is a very expensive procedure. All reported algorithms that calculate typical testors have exponential complexity.

Typical testors have been widely used to evaluate the feature relevance [7] and as support sets in classification algorithms [2]. In text mining, they have also used for text categorization [8] and document summarization [9].

In comparation with another methods for feature selection, Typical testors has a null confusion error, due to do not confuse objetcs of different classes.

Different methods like sequential and parallel algorithms [17] and genetic algorithms used for calculating a subset of typical testors [16] has been developed. Besides, an embedded system based on a FPGA architecture for testor identification [10] was introduced. This architecture allows verify if a feature combination complies the testor property.

But even through the application of these techniques, the run time of existing algorithms continues to be unacceptable owing to several problems which are dependent mainly, of the number of features of training matrix.

The present paper introduces a fast implementation of the CT_EXT algorithm, which simplifies the search of feature combinations which fulfill the testors property.

The classic concept of a testor, in which classes are assumed to be both hard and disjointed, is used. The comparison criteria used for all features are Boolean, regardless of the feature type (qualitative or quantitative). The similarity function used for comparing objects demands similarity in all features. These concepts are formalized in the following section.

2 Basic Concepts

Let TM be a training matrix containing m objects described in terms of n features $R = \{x_1, x_2, \cdots, x_n\}$ and distributed into c classes $\{k_1, k_2, \cdots, k_c\}$. Each feature $x_i \in R$ takes values in a set L_i, $i = 1, \cdots, n$. A comparison criterion of dissimilarity $D : M_i \times M_i \rightarrow \{0, 1\}$ is associated to each x_i (0=similar, 1=dissimilar), where M_i is the admissible values set of x_i.

Applying these comparison criteria for all possible pairs of objects belonging to different classes in TM, a Boolean dissimilarity matrix, denoted by DM, is built.

Notice that the number of rows in DM is

$$m^{'} = \sum_{i=1}^{c-1} \sum_{j=i+1}^{c} card(K_i) \; * \; card(K_j) \tag{1}$$

where $card(k_i)$ denotes the number of objects in the class k_i.

Let p and q be two rows of DM. p is a subrow of q if in all columns where p has 1, q has also it. A row p of DM is called basic if no row in DM is a subrow of p. The submatrix of DM containing all its basic rows (without repetitions) is called a basic matrix (denoted by BM).

Then, a testor is a subset of features $T = \{x_{i_1}, \cdots, x_{i_s}\}$ of TM for which a whole row of zeros does not appear in the remaining submatrix of BM, after eliminating all columns corresponding to the features in $R \backslash T$. T is a typical testor if there is no proper subset of T that meets the testor condition [4]. Commonly, algorithms used for computing typical testors make use of BM instead of DM due to the substantial reduction of rows.

Follow the notation used in [5]. Let $V = (a_1, a_2, \cdots, a_u)$ be a binary u-tuple of elements, $a_i \in \{0, 1\}$, $i = 1, \cdots, u$. We named cardinal of a binary u-tuple, to the number of elements that contains the u-tuple. The column corresponding to a feature $x_i \in BM$ is a binary u-tuple, denoted by V_{x_i}, whose cardinal of this u-tuple is the number of rows in BM.

The logical operations on binary tuples are defined as follows:

$$(a_1, a_2, \cdots, a_u) \vee (b_1, b_2, \cdots, b_u) = (a_1 \vee b_1, a_2 \vee b_2, \cdots, a_u \vee b_u) \qquad (2)$$

$$\neg(a_1, a_2, \cdots, a_u) = (\neg a_1, \neg a_2, \cdots, \neg a_u) \qquad (3)$$

$$\bigoplus(a_1, a_2, \cdots, a_u) = (a_1 \vee a_2 \vee, \cdots, \vee a_u) \qquad (4)$$

$(1, \cdots, 1)$ and $(0, \cdots, 0)$ represent binary tuples in which all elements are one and zero, respectively.

The notation $L = [x_{i_1}, \cdots, x_{i_s}]$, $x_{i_s} \in R$ is used to represent an ordered list of features. A list L does not contain features is denoted as [ ] (empty list).

We call length of a list L, denoted as $len(L)$, to the number of its features. All basic operations of the set theory (difference, intersection, subset or sublist, etc.) can be defined on ordered lists of features in a similar way.

We denote the concatenation between ordered lists of features with the symbol +.

Definition 1. *Let $L = [x_{i_1}, \cdots, x_{i_s}]$ be a feature list. We call accumulative mask of L, denoted as am_L, to the binary tuple in which the i^{th} element is 1 if the i^{th} row in BM has at least a 1 in the columns corresponding to the features of L and it is 0 otherwise.*

Definition 2. *Let $L = [x_{i_1}, \cdots, x_{i_s}]$ be a feature list. We call contribution mask of L in the feature $x_{i_q} \in L$, denoted as $cm_{L,x_{i_q}}$, to the binary tuple in which the i^{th} element is 1 if the i^{th} row in BM has only one 1 in the column corresponding to the feature x_{i_q}, and 0 in the columns belonging to remaining features. And it is 0 otherwise.*

Notice that the cardinal of both am_L and $cm_{L,x_{i_p}}$ is the number of rows in BM.

Example 1. Let $L_1 = [x_2]$, $L_2 = [x_1, x_4]$, $L_3 = [x_1, x_4, x_5]$ and $L_4 = [x_1, x_2, x_3, x_4]$. Its accumulative and contribution masks in the features x_2, x_4, x_5, x_4 are the following: $am_{L_1} = (0, 0, 1, 1)$, $am_{L_2} = (1, 0, 1, 0)$, $am_{L_3} = (1, 1, 1, 0)$, $am_{L_4} = (1, 1, 1, 1)$; $cm_{L_1,x_2} = (0, 0, 1, 1)$, $cm_{L_2,x_4} = (0, 0, 1, 0)$, $cm_{L_3,x_5} = (0, 1, 0, 0)$, $cm_{L_4,x_4} = (0, 0, 0, 0)$.

$$BM = \left(\begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{array}\right) \tag{5}$$

Proposition 1. *Let $L = [x_{i_1}, \cdots, x_{i_s}]$ be a feature list and $x_{i_q} \in R$, $x_{i_q} \notin L$. The accumulative mask of the list $L + [x_{i_q}]$ is calculated as follows:*

$$am_{L+[X_{i_q}]} = am_L \vee V_{x_{i_q}} \tag{6}$$

Proof. We have two cases. a) the binary u-tuple am_L in the i^{th} element has a 0, and $V_{x_{i_q}}$ in its same i^{th} element has a 1. To perform the operation $am_L \vee V_{x_{i_q}}$, this u-tuple in its i^{th} element will now has a 1. b) both the binary u-tuple am_L and $V_{x_{i_q}}$ in its i^{th} element has a 0. In this case, to perform the operation $am_L \vee V_{x_{i_q}}$, this u-tuple in its i^{th} element maintains the value of 0.

For the case in which u-tuple am_L in the i^{th} element has a 1, the value of $V_{x_{i_q}}$ in its i^{th} element is irrelevant, because that performing the operation $am_L \vee V_{x_{i_q}}$, this u-tuple in its i^{th} element retains the value of 1.

Thus, the u-tuple $am_{L+[X_{i_q}]}$ meets definition 1.

Remark 1. The column corresponding to a feature $x_{i_q} \in BM$ is the binary tuple $V_{x_{i_q}}$

Proposition 2. *Let $L = [x_{i_1}, \cdots, x_{i_s}]$ be a feature list and $x_{i_q} \in R$, $x_{i_q} \notin L$. The contribution mask of the list $L + [x_{i_q}]$ is calculated as follows:*

$$cm_{L+[x_{i_q}],x_{i_q}} = \neg am_L \wedge V_{x_{i_q}} \tag{7}$$

Proof. If the binary u-tuple am_L in the i^{th} element has a 0, and $V_{x_{i_q}}$ in its same i^{th} element has a 1, then it is the only case in that to perform the operation $\neg am_L \wedge V_{x_{i_q}}$, this u-tuple in its i^{th} element will now has a 1, according to table 1. For remaining cases, to perform the operation $\neg am_L \wedge V_{x_{i_q}}$, this u-tuple in its i^{th} element will has a 0.

Thus, the u-tuple $cm_{L+[x_{i_q}],x_{i_q}}$ meets definition 2.

Table 1. Value table for contribution mask

B_1	B_2	$\neg B_1$	$\neg B_1 \wedge B_2$
1	1	0	0
1	0	0	0
0	1	1	1
0	0	1	0

Notice that propositions 1 and 2 allow the updating of accumulative and contribution masks, respectively when a new feature is added to a feature list.

Proposition 3. *A feature list $L = [x_{i_1}, \cdots, x_{i_s}]$ is a testor if and only if $am_L = (1, \cdots, 1)$*

Proof. As each element of the u-tuple am_L has a 1, by definition 1, each row of BM has at least a 1 in the columns corresponding to the features of L.

Then, in the columns of the submatrix of BM formed by the features $x_{i_1}, \cdots, x_{i_s}$, there are not a row of zeros. Thus, the features $x_{i_1}, \cdots, x_{i_s}$ form a testor.

Definition 3. *Let $L = [x_{i_1}, \cdots, x_{i_s}]$ be a feature list and $x_{i_q} \in L$. A row p in BM is a typical row of x_{i_q} with respect to L if it has a 1 in the column corresponding to x_{i_q} and zero in all the columns corresponding to the features in $L \backslash [x_{i_q}]$.*

Proposition 4. *A feature list $L = [x_{i_1}, \cdots, x_{i_s}]$ is a typical testor if and only if L is a testor and for every feature $x_{i_q} \in L$ there is at least a typical row of x_{i_q} with respect to L*

Proof. As each feature of L has at least a typical row, then if any feature $x_{i_q} \in L$ is eliminated of L, then a zero row is generated in then remaining features $x_{i_s} \in L \backslash [x_{i_q}]$, $s \neq q$. Thus, $L \backslash [x_{i_q}]$ does not satisfy of being a typical testor.

Therefore, L is a typical testor.

The CT-EXT algorithm [14] has the following theoretical bases.

Definition 4. *Let $L = [x_{i_1}, \cdots, x_{i_s}]$ be a feature list. We say that the row p in BM, denoted it by p_0 is a zero row of L if it has a 0 in all the columns corresponding to the features of L*

Definition 5. *Let $L = [x_{i_1}, \cdots, x_{i_s}]$ be a feature list and $x_{i_q} \in R$, $x_{i_q} \notin L$. We denote the number of zero rows of L by $\sum_L p_0$. We say that x_{i_q} contributes with L if and only if $\sum_{L+[x_{i_q}]} p_0 < \sum_L p_0$*

Proposition 5. *Let $T \subseteq R$ and $x_j \in R$, $x_j \notin T$. If x_j does not contribute to T, then $T \cup \{x_j\}$ can not generate any tipical testor.*

Proposition 6. *Let $T \subseteq R$, $Z \subseteq R$, $Z \neq \emptyset$. If T is a testor, then $T \cup Z$ is a testor too, but is not a typical testor.*

Now, for the fast implementation of the CT_EXT algorithm we have the follow definitions and propositions.

Definition 6. *Let $L = [x_{i_1}, \cdots, x_{i_s}]$ be a feature list and $x_{i_q} \in R$, $x_{i_q} \notin L$. We say that x_{i_q} contributes with L if and only if*

$$\bigoplus cm_{L+[x_{i_q}],x_{i_q}} \neq 0 \qquad (8)$$

Propositions 5 and 6, are re-writen in terms of Definition 6 and Proposition 3, in the following way.

Theorem 1. *Let $L = [x_{i_1}, \cdots, x_{i_s}]$ be a feature list and $x_{i_q} \in R$, $x_{i_q} \notin L$. If x_{i_q} does not contribute with L, then $L + [x_{i_q}]$ can not generate any typical testor.*

Proof. As x_{i_q} does not contribute with L, this means that a) there is not typical row of x_{i_q} with respect to $L+[x_{i_q}]$. Thus, $L+[x_{i_q}]$ does not satisfy proposition 4;

b) it exists the i^{th} element in the accumulative mask of $L + [x_{i_q}]$ which has a 0. And then, the i^{th} row in BM has only zeros in the columns belonging to the features of $L + [x_{i_q}]$. Thus, $L + [x_{i_q}]$ does not satisfy propositions 3 and 4.

However, $L + [x_{i_q}]$ can not generate any typical testor.

Theorem 2. *Let $L = [x_{i_1}, \cdots, x_{i_s}]$ and $Z \subseteq R$, $Z \neq \emptyset$ be a feature lists. If L is a testor, then $L + Z$ is a testor too, but it is not a typical testor.*

Proof. As L is a testor, then $am_L = (1, \cdots, 1)$. Then, no feature of Z contributes with L. Thus, by proposition 1, $L + Z$ is not a typical testor.

And, by proposition 1, the accumulative mask of $L + Z$ is the same that the accumulative mask of L. Then, $L + Z$ is a testor too.

Remark 2.

3 The Fast Implementation of the CT-EXT Algorithm

The algorithm CT_EXT performs a search of features subset on the Basic Matrix obtained from training Matrix. It generates combinations of features, in order to be forming a subset that satisfying the testor property. And then, verify if the typical testor property is fulfilled by the combination generated.

In general, The algorithm CT_EXT works as follows. First, reorders the rows and columns of the Basic Matrix, because CT_EXT is an algorithm which uses the same lexicographic total order that LEX [18] and BR [5] algorithms uses. The CT_EXT algorithm generates incremental feature combinations reducing, step by step, the number of objects belonging to different classes that are confused, until a combination which is a testor is obtained. Subsequently, CT_EXT verifies whether the generated combination is a typical testor. As well as LEX and

BR, CT_EXT rules out those feature combinations that can generate a testor which is not a typical testor, preserving those candidates capable of generating a typical testor only. If a testor is generated, all its consecutive supersets (in the lexicographic order previously introduced into the power set of features) are not analyzed. They are skipped because these feature combinations are testors, but not typical testors.

Now, the fast implementation of the algorithm CT_EXT, is based on that operations of addition and comparison used in CT_EXT, are replaced by simple logical Boolean operations (OR, AND and NOT), to verify if an attribute x_j contributes to a list L (see the Definitions 2 and 6, of which are derived Theorems 1 and 2. And then it is verified, if the list L is a testor.

Thus, the number of operations made by CT_EXT are significantly reduced in the fast implementation of this. And then, the run time of the algorihm is improved significantly.

Description of the fast implementation of the algorithm CT_EXT

```
Input: BM (Basic Matrix)
Output: TT (set of all typical testors)

Step 1: Ordering rows and columns in BM.
The row that have minimum number of 1's, is set as the first row
of BM.
The columns of BM are ordered, from left to right, each having a
value of 1 in the first row and each subsequent column having a
value 0 in the first row of BM. The order of the columns into each
group (with the same value of 1 or with the same value of 0) is
irrelevant.

Step 2: Initializing.
Let the set TT={} (initialized as empty set, it will be the typical
testor set) and the list T=[ ] (empty list, which is the current
feature combination); j=1  (first feature of BM to be analyzed).

Step 3: Adding a new feature of first row of BM.
If Xj has a 1 in the first row of BM, then [Xj] is concatened with
T (T=T + [Xj]), go to step 5.
In another case, the algorithm finishes (any new feature
combination will not generate a typical testor, because all these
feature combinations have a zero row).

Step 4: Evaluating the new feature.
The list [Xj] which contains the feature Xj is concatened to the
current list T (T=T + [Xj]), and it is verified whether this new
feature contributes to the current combination (Definition 6).
If Xj does not contributes with T, then go to step 6.
```

```
Step 5: Verifying testor property.
It verify if the list T is a testor (Proposition 3). If T is a
testor, then verify whether if T is a typical testor (Proposition
4). If T was a typical testor, then T is added to the set TT.
Otherwise, go to step 7.

Step 6: Eliminating the last feature procesed.
The list which contains the last feature procesed Xj is eliminated
from T (T=T \ [Xj]). If Xj does not contribute to T, then no
combination containing T is verified (Theorem 1) and go to
step 7. On the other hand, If the list T was a testor, then no
consecutive concatened list of T is procesed (Theorem 2). If T
is an emptyset, then j=j+1 and go to step 3.

Step 7: Selecting a new feature to analyze.
The next no concatened feature in the current combination is
selected. If j<n then j=j+1, and go to step 4. Otherwise, go to
step 6.
```

4 Experiments

In order to evaluate the performance of the fast implementation for algorithm CT_EXT using the binary accumulative structure, a comparison with four algorithms reported in the literature (BT, CT, LEX and CT_EXT) was made. The first algorithm selected is a classical external type algorithm, which uses the last reported algorithm which incorporates several improvements in performance [15]. The second algorithm is a classical internal type algorithm [13]. The third algorithm, LEX is reported with very well run time execution among classical algorithms [18]. Finally the algorithm CT_EXT, which is other of the fastest algorithms reported too [14].

Please note that, we do not make comparisons with the BR algorithm, and we use a version of the algorithm LEX provided by Dr. Jose Francisco Martinez-Trinidad, from INAOE, Mexico, for fulfill comparisons among algorithms. Because the authors of the algorithms LEX [18] and BR [5] gave us a version of these algorithms in a program (included source code of LEX, but not of the BR algorithm), to perform our comparisons. However, this program does not allowed to read and handled DM or BM previously saved. Only worked with DM and BM generated with random entries.

For compare the run times of the algorithms, we use several BM with different dimensions. Two of them were taken from real medical diagnosis problems. In table 2, the experimental results obtained with the algorithms are shown.

The matrices used are denoted by $M_{rows \times columns}$, and TT denotes the number of typical testors found by algorithms. The experiments were conducted in a Pentium IV, with 2Ghz, and 1 Mbyte of RAM. The execution times are presented

Table 2. Run time execution in seconds of several algorithms

Algorithm	$M_{10\times34}$	$M_{20\times38}$	$M_{209\times32}$	$M_{209\times47}$	$M_{269\times42}$
BT	14	105	25	> 43200	> 43200
CT	0	0	39	8026	38691
LEX	0	0	14	1799	2530
CT_EXT	0	0	3	483	928
FI_CT_EXT	**0**	**0**	**0**	**72**	**120**
TT	935	2,436	6,405	184,920	302,066

in seconds. And the fast implementation of the algorithm CT_EXT is denoted by FI_CT_EXT.

In addition, we evaluate the performance of the algorithm CT_EXT and the fast implementation proposed. We handled four real databases, obtained from UCI Machine Learning Repository [19]. The databases handled are: Zoo database; Mushroom database; Chess (King-Rook vs. King-Pawn), denoted by Kr_vs_Kp; and Molecular Biology (Promoter Gene Sequences), denoted by Promoters. In table 3, experimental results obtained for these databases are shown. In the case of Promoters database, several basic sub-matrices were calculated from the original basic matrix because the dimensions of this last matrix were 2761 rows and 57 columns. Third and fourth columns contain the run time execution of algorithm CT_EXT and the their fast implementation.

In table 3, we can observe that the fast implementation proposed achieves important reductions in the run time execution among 80% and 98% with respect to the algorithm CT_EXT.

Table 3. Run time execution in seconds of the algorithm CT_EXT and their binary extension, handling several real databases

	Zoo	Kr_vs_Kp	Mushroom	Promoters	Promoters	Promoters	Promoters
Dimension of MB rows x columns	14×17	120×35	30×22	100×57	250×57	500×57	1000×57
Run time of CT_EXT	0	295	0	16	167	940	14,979
Run time of binary extension of CT_EXT	**0**	**58**	**0**	**2**	**13**	**47**	**199**
Number of typical testors found	34	8464	292	77,467	257,189	726,700	1,490,107

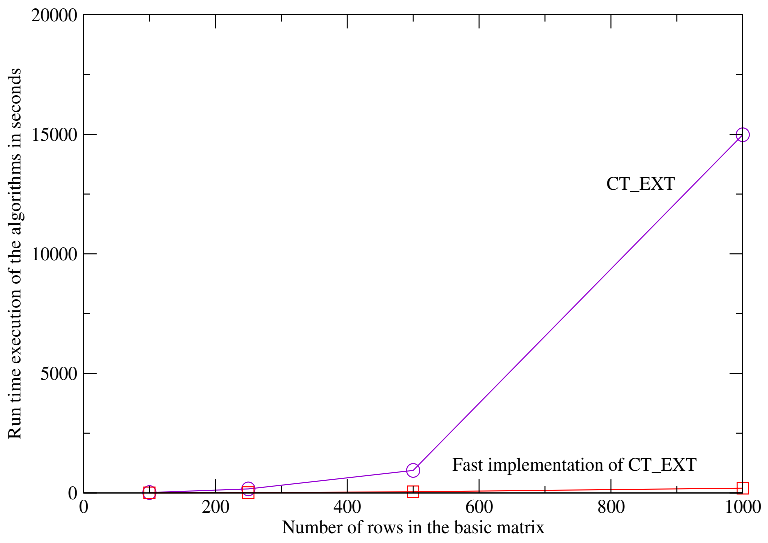


Fig. 1. Run time execution in seconds of CT_EXT algorithm and the fast implementation, when the number of rows is incremented

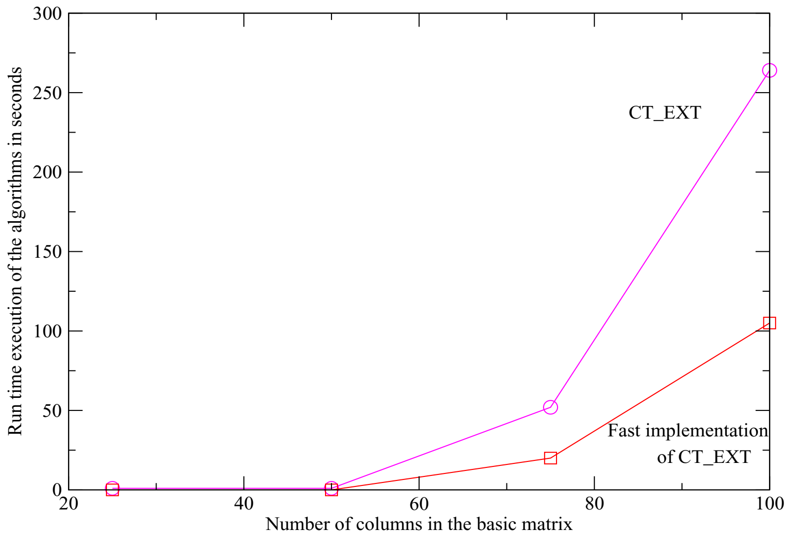


Fig. 2. Run time execution in seconds of CT_EXT algorithm and the fast implementation proposed, when the number of features is increased

In order to study the behavior of the algorithm CT_EXT and their fast implementation, we show in figure 1, the run time of the algorithms for basic matrices of 57 columns varing the number of rows from 100 to 1000. Besides, the run time of these algorithms for basic matrix of 50 rows, but now, varing the number of columns from 25 to 100 is shown in figure 2.

5 Conclusions

In this paper, a fast implementation of the algorithm CT_EXT in order to facilitate identification of testors of a training matrix was proposed.

The fast implementation developed over the algorithm CT_EXT, is feasible to apply in other algorithms which calculated the typical testor set of a training matrix.

The main contribution of the fast implementation, is kept the information of each new feature to process, allowing it to verify fast and easily whether a feature combination fulfills the testor property, because the extension uses simple logical Boolean operations, handles as bits in the code implementation.

Based on experimental results, we can conclude that the fast implementation proposed improves the CT_EXT algorithm.

The fast implementation proposed, does not contemplate the inclusion of noise in BM, because a sensitivity analysis (as is presented in [1]) must be performed, and this study is beyond the scope of the paper.

Besides, we will work to find a criterion for selecting a smaller number of typical testors, when a large amount of typical testor is found.

References

[1] Carrasco-Ochoa, J., Ruiz-Shulcloper, J., Diaz de Leon, J.: Sensitivity analisys in logical combinatorial pattern recognition. Computacion y Sistemas 6(1), 62–66 (2002)

[2] De la Vega-Doria, L., Carrasco-Ochoa, A., Ruiz-Shucloper, J.: Fuzzy KORA-W algorithm. In: 6th European Conf. on Intelligent Techniques and Soft Computer, Germany, pp. 1190–1194 (1998)

[3] Dmitriev, A.N., Zhuravlev, Y.I., Krendeliev, F.: About mathematical principles and phenomena classification. Diskretni Analiz. 7, 3–15 (1966)

[4] Lazo-Cortes, M., Ruiz-Shulcloper, J., Alba-Cabrera, E.: An Overview of the evolution of the concept of testor. Pattern Recognition 34(4), 753–762 (2001)

[5] Lias-Rodriguez, A., Pons-Porrata, A.: BR: A new method for computing all typical testors. In: Bayro-Corrochano, E., Eklundh, J.-O. (eds.) CIARP 2009. LNCS, vol. 5856, pp. 433–440. Springer, Heidelberg (2009)

[6] Martinez-Trinidad, J.F., Guzman-Arenas, A.: The Logical Combinatorial approach for pattern recognition an overview through selected Works. Pattern Recognition 34(4), 741–751 (2001)

[7] Ortiz-Posadas, M., Martinez-Trinidad, J., Ruiz-Shulcloper, J.: A new approach to diferential diagnosis of diseases. International Journal of Biomedical Computing 40(3), 179–185 (2001)

[8] Pons-Porrata, A., Gil-Garcia, R., Berlanga-Llavori, R.: Using Typical Testors for Feature Selection in Text Categorization. In: Rueda, L., Mery, D., Kittler, J. (eds.) CIARP 2007. LNCS, vol. 4756, pp. 643–652. Springer, Heidelberg (2007)

[9] Pons-Porrata, A., Ruiz-Shulcloper, J., Berlanga-Llavori, R.: A Method for the Automatic Summarization of Topic-Based Clusters of Documents. In: Sanfeliu, A., Ruiz-Shulcloper, J. (eds.) CIARP 2003. LNCS, vol. 2905, pp. 596–603. Springer, Heidelberg (2003)

[10] Rojas, A., Cumplido, R., Carrasco, A., Feregrino, C., Martinez, J.: On the design and implementation of a high performance configurable architecture for testor identification. In: Yin, H., Tino, P., Corchado, E., Byrne, W., Yao, X. (eds.) IDEAL 2007. LNCS, vol. 4881, pp. 665–673. Springer, Heidelberg (2007)
[11] Ruiz, J., Guzman, A., Martinez, J.: Logical combinatorial pattern recognition approach. In: Advances on pattern recognition series, Edit. Instituto Politecnico Nacional, Mexico (1999)
[12] Ruiz-Shulcloper, J., Abidi, M.: Logical Combinatorial Pattern Recognition: A Review. In: Pandalai, S.G. (ed.) Recent Research Developments in Pattern Recognition, Transworld Research Networks, Kerala, India, pp. 133–176 (2002)
[13] Sanchez-Diaz, G.: Developing and implementing efficient algorithms (bath and parallel) for calculating typical testors of a basic matrix. Master Thesis. Autonomous University of Puebla, Puebla, Mexico (1997)
[14] Sanchez-Diaz, G., Lazo-Cortes, M.: CT_EXT: an external escale algorithm for generated typical testors. In: Rueda, L., Mery, D., Kittler, J. (eds.) CIARP 2007. LNCS, vol. 4756, pp. 506–514. Springer, Heidelberg (2007)
[15] Sanchez-Diaz, G., Lazo-Cortes, M.: Modifications to BT algorithm for improving its run time execution. Revista Ciencias Matematicas 20(2), 129–136 (2002)
[16] Sanchez-Diaz, G., Lazo-Cortes, M., Fuentes-Chavez, O.: Genetic algorithm for calculating typical testors of minimal cost. In: Iberoamerican Symposium on Pattern Recognition, pp. 207–213 (1999)
[17] Sanchez-Diaz, G., Lazo-Cortes, M., Garcia-Fernandez, J.: Parallel and distributed models for calculating typical testors. In: Iberoamerican Workshop on Pattern Recognition, pp. 135–140 (1997)
[18] Santiesteban-Alganza, Y., Pons-Porrata, A.: LEX: A new algorithm for calculating typical testors. Revista Ciencias Matematicas 21(1), 85–95 (2003)
[19] UCI-Machine-Learning-Repository, University of California, http://archive.ics.uci.edu/ml

Supervised Probabilistic Classification Based on Gaussian Copulas

Rogelio Salinas-Gutiérrez, Arturo Hernández-Aguirre, Mariano J.J. Rivera-Meraz, and Enrique R. Villa-Diharce

Center for Research in Mathematics (CIMAT), Guanajuato, México
{rsalinas,artha,mrivera,villadi}@cimat.mx

Abstract. This paper introduces copula functions and the use of the Gaussian copula function to model probabilistic dependencies in supervised classification tasks. A copula is a distribution function with the implicit capacity to model non linear dependencies via concordance measures, such as Kendall's τ. Hence, this work studies the performance of a simple probabilistic classifier based on the Gaussian copula function. Without additional preprocessing of the source data, a supervised pixel classifier is tested with a 50-images benchmark; the experiments show this simple classifier has an excellent performance.

Keywords: Gaussian copula, supervised classification.

1 Introduction

In Pattern Recognition applications many algorithms and models have been proposed for many tasks, specially for *clustering*, *regression* and *classification*. Applications in which a *training data set* with categories and attributes is available and the goal is to assign a new object to one of a finite number of discrete categories are known as *supervised classification* problems [2,12,15]. In this work we present the use of the Gaussian copula function as an alternative for modeling dependence structure in a supervised probabilistic classifier.

Copula functions are suitable tools in statistics for modeling multiple dependence, not necessarily linear dependence, in several random variables. For this reason, copula functions have been widely used in economics and finance [5,7,9,26,27]. More recently copula functions have been used in other fields such as climate [23], oceanography [6], hydrology [10], geodesy [1], reliability [17], evolutionary computation [21,22] and engineering [11]. By using copula theory, a joint distribution can be built with a copula function and, possibly, several different marginal distributions. Copula theory has been used also for modeling multivariate distributions in *unsupervised learning* problems such as image segmentation [4,8] and retrieval tasks [16,20,25]. In [13], the bivariate copula functions Ali-Mikhail-Haq, Clayton, Frank and Gumbel are used for unsupervised classification. These copulas are well defined for two variables but when extended to three or more variables several complications arise (for instance, undefined copula parameters), preventing their generalization and applicability.

For the Gaussian copula however, there exist a simple method for any number of variables. This work introduces the use of Gaussian copula in supervised classification, and compares an independent probabilistic classifier with a copula-based probabilistic classifier.

The content of the paper is the following: Section 2 is a short introduction to copula functions, Section 3 presents a copula based probabilistic model for classification. Section 4 presents the experimental setting to classify an image database, and Section 5 summarizes the conclusions.

2 Copula Functions

The copula concept was introduced 50 years ago by Sklar [24] to separate the effect of dependence from the effect of marginal distributions in a joint distribution. Although copula functions can model linear and nonlinear dependencies, they have been barely used in computer science applications where nonlinear dependencies are common and need to be represented.

Definition 1. *A copula C is a joint distribution function of standard uniform random variables. That is,*

$$C(u_1, \ldots, u_d) = P(U_1 \leq u_1, \ldots, U_d \leq u_d) \ ,$$

where $U_i \sim U(0,1)$ for $i = 1, \ldots, d$.

For a more formal definition of copula functions, the reader is referred to [14,18]. The following result, known as Sklar's theorem, states how a copula function is related to a joint distribution function.

Theorem 1 (Sklar's theorem). *Let F be a d-dimensional distribution function with marginals $F_1, F_2, \ldots, F_d$, then there exists a copula C such that for all x in $\overline{\mathbb{R}}^d$,*

$$F(x_1, x_2, \ldots, x_d) = C(F_1(x_1), F_2(x_2), \ldots, F_d(x_d)) \ ,$$

where $\overline{\mathbb{R}}$ denotes the extended real line $[-\infty, \infty]$. If $F_1(x_1)$, $F_2(x_2)$, $\ldots$, $F_d(x_d)$ are all continuous, then C is unique. Otherwise, C is uniquely determined on $Ran(F_1) \times Ran(F_2) \times \cdots \times Ran(F_d)$, where Ran stands for the range.

According to Theorem 1, any joint distribution function F with continuous marginals $F_1, F_2, \ldots, F_d$ has associated a copula function C. Moreover, the associated copula C is a function of the marginal distributions $F_1, F_2, \ldots, F_d$. An important consequence of Theorem 1 is that the d-dimensional joint density f and the marginal densities $f_1, f_2, \ldots, f_d$ are also related:

$$f(x_1, \ldots, x_d) = c(F_1(x_1), \ldots, F_d(x_d)) \cdot \prod_{i=1}^{d} f_i(x_i) \ , \qquad (1)$$

where c is the density of the copula C. The Equation (1) shows that the product of marginal densities and a copula density builds a d-dimensional joint density.

Notice that the dependence structure is given by the copula function and the marginal densities can be of different distributions. This contrasts with the usual way to construct multivariate distributions, which suffers from the restriction that the marginals are usually of the same type. The separation between marginal distributions and a dependence structure explains the modeling flexibility given by copula functions.

2.1 Gaussian Copula Function

There are several parametric families of copula functions, such as Student's t copula and Archimedean copulas. One of these families is the Gaussian copula function.

Definition 2. *The copula associated to the joint standard Gaussian distribution is called Gaussian copula.*

According to Definition 2 and Theorem 1, if the d-dimensional distribution of a random vector $(Z_1, \ldots, Z_d)$ is a joint standard Gaussian distribution, then the associated Gaussian copula has the following expression:

$$C(\Phi(z_1), \ldots, \Phi(z_d); \Sigma) = \int_{-\infty}^{z_1} \cdots \int_{-\infty}^{z_d} \frac{e^{-\frac{1}{2}t'\Sigma^{-1}t}}{(2\pi)^{(n/2)}|\Sigma|^{1/2}} dt_d \cdots dt_1 \ ,$$

or equivalently,

$$C(u_1, \ldots, u_d; \Sigma) = \int_{-\infty}^{\Phi^{-1}(u_1)} \cdots \int_{-\infty}^{\Phi^{-1}(u_d)} \frac{e^{-\frac{1}{2}t'\Sigma^{-1}t}}{(2\pi)^{(n/2)}|\Sigma|^{1/2}} dt_d \cdots dt_1 \ ,$$

where Φ is the cumulative distribution function of the marginal standard Gaussian distribution and Σ is a symmetric matrix with main diagonal of ones. The elements outside the main diagonal of matrix Σ are the pairwise correlations ρ_{ij} between variables Z_i and Z_j, for $i, j = 1, \ldots, d$ and $i \neq j$. It can be noticed that a d-dimensional standard Gaussian distribution has mean vector zero and a correlation matrix Σ with $d(d-1)/2$ parameters.

The dependence parameters ρ_{ij} of a d-dimensional Gaussian copula can be estimated using the maximum likelihood method. To do so, we follow the steps of Algorithm 1.

Algorithm 1. Pseudocode for estimating parameters

```
1: for each random variable $X_i$, $i = 1, \ldots, d$, estimate its marginal distribution function $\hat{F}_i$ using the observed values $x_i$. The marginal distribution function can be parametric or nonparametric
2: determine $u_i = \hat{F}_i(x_i)$, for $i = 1, \ldots, d$
3: calculate $z_i = \Phi^{-1}(u_i)$ where $\Phi$ is the cumulative standard Gaussian distribution function, for $i = 1, \ldots, d$
4: estimate the correlation matrix $\hat{\Sigma}$ for the random vector $(Z_1, \ldots, Z_d)$ using pseudo observations $(z_1, \ldots, z_d)$
```

Due to Equation (1), the d-dimensional Gaussian copula density can be calculated as:

$$c(\Phi(z_1),\ldots,\Phi(z_d);\Sigma) = \frac{\frac{1}{(2\pi)^{(d/2)}|\Sigma|^{1/2}}e^{-\frac{1}{2}z'\Sigma^{-1}z}}{\prod_{i=1}^{d}\frac{1}{(2\pi)^{1/2}}e^{-\frac{1}{2}z_i^2}} = \frac{1}{|\Sigma|^{1/2}}e^{-\frac{1}{2}z'(\Sigma^{-1}-I)z} \quad . \tag{2}$$

Given that a Gaussian copula is a distribution function it is possible to simulate data from it. The main steps are the following: once a correlation matrix Σ is specified, a data set can be generated from a joint standard Gaussian distribution. The next step consists of transforming this data set using the cumulative distribution function Φ. For random vectors with a Gaussian copula associated to their joint distribution, the first step is to generate data from the copula and then determining their quantiles by means of their cumulative distribution functions. Algorithm 2 illustrates the sampling procedure for different correlations.

Algorithm 2. Pseudocode for generating data with Gaussian dependence structure

```
1: simulate observations $(z_1,\ldots,z_d)$ from a joint standard Gaussian distribution with matrix correlation $\Sigma$
2: calculate $u_i = \Phi(z_i)$ where $\Phi$ is the cumulative standard Gaussian distribution function, for $i = 1,\ldots,d$
3: determine $x_i$ using quasi-inverse $F_i^{-1}(u_i)$, where $F_i$ is a cumulative distribution function, for $i = 1,\ldots,d$
```

An important result (see [18]) for parametric bivariate copulas relates the dependence parameter θ to Kendall's τ:

$$\tau(X_1,X_2) = 4\int_0^1\int_0^1 C(u_1,u_2;\theta)dC(u_1,u_2;\theta) - 1 \quad . \tag{3}$$

For a bivariate Gaussian copula, Equation (3) can be written as

$$\tau = \frac{2}{\pi}\arcsin(\rho) \quad . \tag{4}$$

Given that is well established how to estimate correlation matrices, evaluate densities, and calculate integrals for the multidimensional Gaussian distribution, the Gaussian copula function is relatively easy to implement.

3 The Probabilistic Classifier

As noted, the aim of this work is to introduce the use of Gaussian copula functions in supervised classification. According to Theorem 1, we can employ a copula function in a probabilistic classifier, such as a Bayessian classifier. In this section we present a three dimensional probabilistic model based on three empirical distribution functions and a trivariate dimensional Gaussian copula function.

The Bayes' theorem states the following:

$$P(K=k|E=e) = \frac{P(E=e|K=k) \times P(K=k)}{P(E=e)} \ , \tag{5}$$

where $P(K = k|E = e)$ is the posterior probability, $P(E = e|K = k)$ is the likelihood function, $P(K = k)$ is the prior probability and $P(E = e)$ is the data probability.

The Equation (5) has been used as a tool in supervised classification. A probabilistic classifier can be designed comparing the posterior probability that an object belongs to class K given its attributes E. The object is then assigned to the class with the highest posterior probability. For practical reasons, the data probability $P(E)$ does not need to be evaluated for comparing posterior probabilities. Furthermore, the prior probability $P(K)$ can be substituted by an uniform distribution if the user does not have an informative distribution.

3.1 The Probabilistic Classifier Based on Gaussian Copula Function

For continuous attributes, a Gaussian copula function can be used for modeling the dependence structure in the likelihood function. In this case, the Bayes' theorem can be written as:

$$P(K=k|e) = \frac{c(F_1(e_1), \ldots, F_n(e_n)|\Sigma, k) \times \prod_{i=1}^{n} f_i(e_i|k) \times P(K=k)}{f(e_1, \ldots, e_n)} \ , \tag{6}$$

where F_i and f_i are the marginal distribution functions and the marginal densities of attributes, respectively. The function c is a d-dimensional Gaussian copula density defined by Equation (2). As can be seen in Equation (6), each category determines a likelihood function.

3.2 The Probabilistic Classifier Based on Independent Model

By considering conditional independence among the attributes in Equation (6), or equivalently, an independent structure in the likelihood function given a category, a probabilistic classifier can use the following expression in order to calculate posterior probabilities:

$$P(K=k|e) = \frac{\prod_{i=1}^{n} f_i(e_i|k) \times P(K=k)}{f(e_1, \ldots, e_n)} \ . \tag{7}$$

Equation (7) uses an independent structure given by a copula density equals to one. This independent copula density can be also obtained by a Gaussian copula function when matrix Σ is the identity matrix I.

3.3 An Application Example

Consider the following specific classification problem: assign a pixel to a certain class according to its color attributes. If we have information about the color distribution of each class, then we can use this information and the Bayes' theorem in order to classify new pixels. This is an example of supervised classification. For a red-green-blue (RGB) color space and two classes, a Gaussian copula based classifier can be written as

$$P(k|r,g,b) = \frac{c(F_R(r), F_G(g), F_B(b)|\Sigma, k) f_R(r|k) f_G(g|k) f_B(b|k) \times P(k)}{f(r,g,b)} \quad , \quad (8)$$

where c is a trivariate Gaussian copula density.

In order to classify a pixel, we use in Equation (8) a prior probability $P(K = k)$ based on the uniform distribution, nonparametric marginal densities $\hat{f}$ based on histograms to approximate $f_R(r|k)$, $f_G(g|k)$ and $f_B(b|k)$, and nonparametric marginal distributions $\hat{F}$ based on empirical cumulative distribution functions to approximate $F_R(r)$, $F_G(g)$ and $F_B(b)$. For modeling the dependence structure of the likelihood function $f(r,g,b|k)$ we present the use of a trivariate Gaussian copula function.

4 Experiments

We use two probabilistic models in order to classify pixels of 50 test images. The first model is an independent probabilistic model (I-M) based on the product of marginal distributions. The second model is a copula-based model (GC-M) that takes into account a dependence structure by means of a trivariate Gaussian copula. The image database was used in [3] and is available online [19]. This image database provides information about two classes: the foreground and the background. The training data and the test data are contained in the labelling-lasso files [19], whereas the correct classification is contained in the segmentation

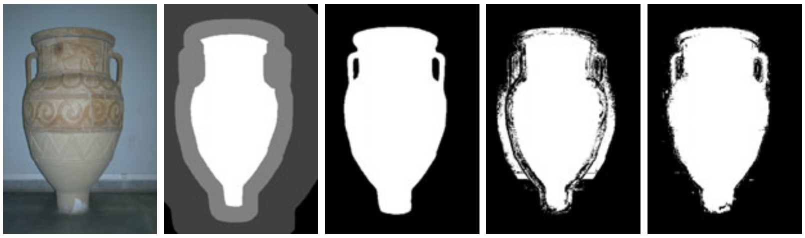


Fig. 1. (a) The color image. (b) The labelling-lasso image with the training data for background (dark gray), for foreground (white) and the test data (gray). (c) The correct classification with foreground (white) and background (black). (d) Classification made by I-M. (e) Classification made by GC-M.

		Truth	
		Positive	Negative
Model	Positive	tp	fp
	Negative	fn	tn

(a)

$$accuracy = \frac{tp + tn}{tp + fp + fn + tn}$$

$$sensitivity = \frac{tp}{tp + fn}$$

$$specificity = \frac{tn}{tn + fp}$$

(b)

Fig. 2. (a) A confusion matrix for binary classification, where tp are true positive, fp false positive, fn false negative, and tn true negative counts. (b) Definitions of accuracy, sensitivity and specificity used in this work.

files. Figure 1 shows the description of one image from the database. Table 3 shows a description for each image. Although the database is used for segmentation purposes, the aim of this work is to introduce the use of the Gaussian copula function in supervised color pixel classification. We use the information for supervised color pixel classification, without taking into account the spatial information.

Three evaluation measures are used in this work: *accuracy*, *sensitivity* and *specificity*. These measures are described in Figure 2. The sensitivity and specificity measures explain the percentage of well classified pixels for each class, foreground and background, respectively. We define the positive class as foreground and the negative class as background.

4.1 Numerical Results

In Table 1 we summarize the measure values reached by the independent probabilistic model (I-M) and the copula-based model (GC-M). The information about the number of pixels well classified for each class is reported in Table 3.

Table 1. Descriptive results for all evaluation measures. BG stands for the background class and FG stands for the foreground class.

Measure	Minimum	Median	Mean	Maximum	Std. deviation
I-M					
Specificity – BG	0.404	0.857	0.817	0.993	0.133
Sensitivity – FG	0.395	0.763	0.768	1.000	0.172
Accuracy	0.571	0.792	0.795	0.976	0.107
GC-M					
Specificity – BG	0.551	0.924	0.885	0.994	0.108
Sensitivity – FG	0.484	0.875	0.854	0.998	0.127
Accuracy	0.587	0.889	0.871	0.987	0.083

Table 2. Results for the difference between evaluation measure means in each model. A 95% confidence interval and a p-value are obtained through a Bootstrap technique. BG stands for the background class and FG stands for the foreground class.

Measure	95% Interval		p-value
Specificity – BG	-1.16E-01	-2.19E-02	6.50E-03
Sensitivity – FG	-1.44E-01	-2.63E-02	6.17E-03
Accuracy	-1.14E-01	-3.98E-02	6.67E-05

To properly compare the performance of the probabilistic models, we conducted a hypothesis test based on a Bootstrap method for the differences between the means of each evaluation measure, for both probabilistic models. Table 2 shows the confidence interval for the means, and the corresponding p-value.

4.2 Discussion

According to Table 1, the GC-M shows the best behaviour for all evaluation measures. For instance, the mean accuracy for the I-M, 79.5%, is less than the mean accuracy for the GC-M, 87.1%. This means that using a I-M approximately has 8% more error rate than using a GC-M.

The average of the specificity is greater than the average of the sensitivity, for both I-M and GC-M (see Table 1). In average, according to Table 1, the GC-M improves the I-M in both classes. For the foreground class from 76.8% to 85.4%, and for the background class from 81.7% to 88.5%.

Table 1 also shows information about the standard deviations for each evaluation measure. For all cases, the standard deviation indicates that using a GC-M in pixel classification is more consistent than using an I-M.

In order to statistically compare the performance of the probabilistic models, Table 2 shows confidence intervals and p-values that confirm differences between the models. None of confidence intervals include the 0 value and all p-values are less than $\alpha = 0.05$.

5 Conclusions

In this work we introduce the use of Gaussian copulas in supervised pixel classification. According to numerical experiments the selection of a Gaussian copula for modeling structure dependence can help achieve better classification results. An specific example is the image *227092*, which appears in Figure 1, its accuracy for the I-M classifier is 57.1%, whereas its accuracy for the GC-M classifier is 89.5%. For this image, the Gaussian copula improves its accuracy.

Although we model the dependence structure for each image with the same copula function, this is not necessary. There are many copula functions and the Gaussian copula has been chosen due to its practical usefulness and easy implementation. However, having more than one copula at hand may improve the performance of the copula-based classifier. In such case, a copula selection procedure is necessary. The evaluation results are the consequence of the selected dependence structure and marginals. For instance, on the image *106024*, the performance of the I-M classifier is 57.6% accurate (accuracy), whereas the GC-M classifier is 58.7% accurate. For most applications better results can be obtained by selecting the best fitted copula function from a set of available copulas. For example, in the experiment reported, the performance of the I-M classifier is better than GC-M for image *fullmoon*. However, the copula based model is expected to improve the performance of the I-M classifier if we used the proper copula.

References

1. Bacigál, T., Komorníková, M.: Fitting archimedean copulas to bivariate geodetic data. In: Rizzi, A., Vichi, M. (eds.) Compstat 2006 Proceedings in Computational Statistics, pp. 649–656. Physica-Verlag HD, Heidelberg (2006)
2. Bishop, C.: Pattern Recognition and Machine Learning. Information Science and Statistics. Springer, Heidelberg (2007)
3. Blake, A., Rother, C., Brown, M., Perez, P., Torr, P.: Interactive image segmentation using an adaptive gmmrf model. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3021, pp. 428–441. Springer, Heidelberg (2004)
4. Brunel, N., Pieczynski, W., Derrode, S.: Copulas in vectorial hidden markov chains for multicomponent image segmentation. In: ICASSP 2005: Proceedings of the 2005 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 717–720 (2005)
5. Cherubini, U., Luciano, E., Vecchiato, W.: Copula Methods in Finance. Wiley, Chichester (2004)
6. De-Waal, D., Van-Gelder, P.: Modelling of extreme wave heights and periods through copulas. Extremes 8(4), 345–356 (2005)
7. Dowd, K.: Copulas in macroeconomics. Journal of International and Global Economic Studies 1(1), 1–26 (2008)
8. Flitti, F., Collet, C., Joannic-Chardin, A.: Unsupervised multiband image segmentation using hidden markov quadtree and copulas. In: IEEE International Conference on Image Processing. Genova, Italy (September 2005), `http://www.icip05.org/`
9. Frees, E.W., Valdez, E.A.: Understanding relationships using copulas. North American Actuarial Journal 2(1), 1–25 (1998)
10. Genest, C., Favre, A.: Everything you always wanted to know about copula modeling but were afraid to ask. Journal of Hydrologic Engineering 12(4), 347–368 (2007)
11. Grigoriu, M.: Multivariate distributions with specified marginals: Applications to wind engineering. Journal of Engineering Mechanics 133(2), 174–184 (2007)
12. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edn. Springer Series in Statistics. Springer, Heidelberg (2009)
13. Jajuga, K., Papla, D.: Copula functions in model based clustering. In: From Data and Information Analysis to Knowledge Engineering. Proceedings of the 29th Annual Conference of the Gesellschaft für Klassifikation e.V., pp. 606–613. Springer, Heidelberg (2006)

14. Joe, H.: Multivariate models and dependence concepts. Chapman & Hall, London (1997)
15. Mackay, D.: Information Theory, Inference, and Learning Algorithms, Cambridge (2008)
16. Mercier, G., Bouchemakh, L., Smara, Y.: The use of multidimensional copulas to describe amplitude distribution of polarimetric sar data. In: IGARSS 2007 (2007)
17. Monjardin, P.: Análisis de dependencia en tiempo de falla. Master's thesis, Centro de Investigación en Matemáticas, Guanajuato, México (December 2007) (in Spanish)
18. Nelsen, R.: An Introduction to Copulas, 2nd edn. Springer Series in Statistics. Springer, Heidelberg (2006)
19. Rother, C., Kolmogorov, V., Blake, A., Brown, M.: Image and video editing, http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm
20. Sakji-Nsibi, S., Benazza-Benyahia, A.: Multivariate indexing of multichannel images based on the copula theory. In: IPTA 2008 (2008)
21. Salinas-Gutiérrez, R., Hernández-Aguirre, A., Villa-Diharce, E.: Using copulas in estimation of distribution algorithms. In: Hernández Aguirre, A., Monroy Borja, R., Reyes García, C. (eds.) MICAI 2009. LNCS (LNAI), vol. 5845, pp. 658–668. Springer, Heidelberg (2009)
22. Salinas-Gutiérrez, R., Hernández-Aguirre, A., Villa-Diharce, E.: D-vine eda: a new estimation of distribution algorithm based on regular vines. In: GECCO 2010: Proceedings of the 12th annual conference on Genetic and evolutionary computation, pp. 359–366. ACM, New York (2010)
23. Schölzel, C., Friederichs, P.: Multivariate non-normally distributed random variables in climate research – introduction to the copula approach. Nonlinear Processes in Geophysics 15(5), 761–772 (2008), http://www.nonlin-processes-geophys.net/15/761/2008/
24. Sklar, A.: Fonctions de répartition à n dimensions et leurs marges. Publications de l'Institut de Statistique de l'Université de Paris 8, 229–231 (1959)
25. Stitou, Y., Lasmar, N., Berthoumieu, Y.: Copulas based multivariate gamma modeling for texture classification. In: ICASSP 2009: Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1045–1048. IEEE Computer Society, Washington (2009)
26. Trivedi, P., Zimmer, D.: Copula Modeling: An Introduction for Practitioners, Foundations and Trends® in Econometrics, vol. 1. Now Publishers (2007)
27. Venter, G., Barnett, J., Kreps, R., Major, J.: Multivariate copulas for financial modeling. Variance 1(1), 103–119 (2007)

Appendix

Table 3. Description of images used in this work. BG stands for the background class and FG stands for the foreground class. Columns 3 and 4 give the size of test data. The last 4 columns give the number of pixels well classified for each class and for each probabilistic classifier.

Image name	Image size	Test pixels		I-M		GC-M	
		BG	FG	BG	FG	BG	FG
21077	321 × 481	4322	3353	3648	2551	3144	2763
24077	321 × 481	11529	10983	6577	8399	6348	10000
37073	321 × 481	8260	6642	6404	6187	7115	6014
65019	321 × 481	9853	8398	9181	3317	9099	4061
69020	321 × 481	25203	22634	17561	16813	22798	20421
86016	321 × 481	3271	2215	2765	2166	2937	2179
106024	321 × 481	9093	7368	5528	3961	5574	4087
124080	321 × 481	18286	18773	16487	16924	16307	18653
153077	321 × 481	13851	12098	11072	7774	10806	10638
153093	321 × 481	12027	11809	7617	8699	11414	9615
181079	481 × 321	23845	23110	18650	15320	22494	18705
189080	481 × 321	23363	23523	20726	21020	19722	20707
208001	481 × 321	10227	9530	9994	7669	10064	7914
209070	321 × 481	6696	4075	5117	2447	5894	2874
227092	481 × 321	19656	17321	12869	8229	19129	13966
271008	321 × 481	10909	9216	8934	7967	8800	8795
304074	481 × 321	7239	4794	5017	2591	5534	2810
326038	321 × 481	10781	7680	8730	4952	9488	5571
376043	481 × 321	13654	13485	12022	6094	13072	9343
388016	481 × 321	17800	15592	15633	11248	17596	12929
banana1	480 × 640	29983	24052	17120	23964	20285	23601
banana2	480 × 640	27433	21518	17063	20378	25373	18698
banana3	480 × 640	26205	20164	25588	12405	26035	14115
book	480 × 640	26087	21474	15689	20699	19852	21325
bool	450 × 520	20123	16850	19500	13279	18726	14373
bush	600 × 450	32513	22099	21072	12504	27734	14870
ceramic	480 × 640	30549	25709	24809	25069	27328	24791
cross	600 × 450	34602	25733	32824	25703	32918	25132
doll	549 × 462	18866	15106	12976	13269	17947	14960
elefant	480 × 640	27858	22787	20918	22656	23158	22540
flower	450 × 600	16125	13246	14612	12977	15036	13225
fullmoon	350 × 442	1580	1043	1498	1043	983	1026
grave	600 × 450	12294	12832	11977	11567	12219	10889
llama	371 × 513	8930	8445	7783	5322	7547	7287
memorial	600 × 450	14853	12598	12900	6902	10936	10964
music	480 × 640	23945	19494	20457	18723	21794	19112
person1	450 × 600	19092	16384	16452	10041	18831	15372
person2	450 × 600	12796	9595	11492	5358	12465	9219
person3	600 × 450	14649	11450	13494	8122	14022	10112

Table 3. (*continued*)

person4	450 × 600	19250	16631	15230	10691	18197	11653
person5	600 × 450	13990	11332	13009	8327	13025	10377
person6	600 × 450	19015	15645	16753	9038	16071	11732
person7	600 × 450	12110	9634	9934	6998	11795	8093
person8	480 × 640	16684	12741	6740	11157	14690	9534
scissors	480 × 640	30768	23335	28152	19910	30181	19960
sheep	600 × 450	5331	3733	4750	3098	5243	3415
stone1	480 × 640	18716	15635	16087	15525	18376	15528
stone2	480 × 640	22002	18489	21556	16315	21788	17692
teddy	398 × 284	13892	13739	13790	13191	13426	13466
tennis	472 × 500	19471	13129	18054	8673	18322	8613

Text-Independent Speaker Identification Using VQ-HMM Model Based Multiple Classifier System

Ali Zulfiqar[1], Aslam Muhammad[2], A.M. Martinez-Enriquez[3], and G. Escalada-Imaz[4]

[1] Departament of CS & IT, University of Gujrat, Pakistan
zulfiqar.butt@uog.edu.pk
[2] Departament of CS & E, U. E. T., Lahore, Pakistan
maslam@uet.edu.pk
[3] Department of Computer Science, CINVESTAV-IPN, Mexico
ammartin@cinvestav.mx
[4] Artificial Intelligence Research Institute, CSIC, Barcelona, Spain
gonzalo@iiia.csic.es

Abstract. Every feature extraction and modeling technique of voice/speech is not suitable in all type of environments. In many real life applications, it is not possible to use all type of feature extraction and modeling techniques to design a single classifier for speaker identification tasks because it will make the system complex. So instead of exploring more techniques or making the system complex it is more reasonable to develop the classifier by using existing techniques and then combine them by using different combination techniques to enhance the performance of the system. Thus, this paper describes the design and implementation of a VQ-HMM based Multiple Classifier System by using different combination techniques. The results show that the developed system by using confusion matrix significantly improve the identification rate.

Keywords: Speaker identification, classifier combination, HMM, VQ, MFCC, LPC.

1 Introduction

Speaker identification (SI) process identifies an unknown registered speaker by comparing it with those registered speaker voice stored in the database. SI can be text-dependent and text-independent [1]. Text-independent SI system is not limited to recognize speakers on the basis of same sentences stored in the database. While text-dependent SI system only can recognize speakers by uttering the same sentence every time [2]. SI can be further divided into closed set SI and open set SI [3]. In closed set speaker identification, unknown speech signal came from one of the registered speakers. Open-set speaker identify unknown signal from either the set of the registered speakers or unregistered speakers. Closed-set text-independent speaker identification (CISI) system must allow capturing particular voice features even in a noisy environment.

Two widely used feature extraction techniques - Mel-frequency Cepstral Coefficients (MFCC) [4] and Linear Prediction Coefficients (LPC) [5], and two modeling techniques – Hidden Markov model (HMM) [6] and Vector Quantization (VQ) [7] are used to construct different classifiers Each classifier is different from each other, in feature extraction and the modeling techniques used. MFCC simulates the behavior of human ear and uses Mel Frequency scale. LPC features represent the main vocal tract resonance property in the acoustic spectrum and make possible to distinguish one speaker from others, due to each speaker is characterized by his/her own formant structure. HMM based on Markov chain mathematical model is a doubly stochastic process that recognizes speakers very well in both text-dependent and text-independent SI system. VQ is implemented through LBG algorithm to reduce and compress feature vectors into a small number of highly representative vectors.

The speaker identification made by a single decision making scheme is always a risky because each type of features are not suitable for all environments. Thus, this paper describes a Multiple Classifier System (MCS) for CISI which reduces errors and wrong identification. The basic idea is to analyze the results obtained by different classifiers. Then, these classifiers are integrated such that their reliability is enhanced due to a proper combination technique. The principle objective of this work is to get the better identification rate of MCS for CISI by using various combination techniques to compound the output of individual classifiers.

The paper proceeds as follows: Section 2 describes the steps followed by all three developed classifiers for the SI. Section 3 explains the different combination techniques used in MCS to coalesce the normalized measurement level output of single classifiers for the joint decision. Section 4 depicts the results obtained from system testing and experimentations. Finally, we conclude our work in Section 5.

2 Single Classifier Speaker Identification System

Three different classifiers are designed and implemented: - LPC based on Vector Quantization (VQ) (classifier K1); - MFCC based VQ (classier K2); and - MFCC based on HMM (classifier K3), (see Figure 1). All three classifiers are able to perform the closed-set text-independent speaker identification. Each classifier of any SI task includes following steps [8], [9], [10]:

- *Digital speech data acquisition.* Acoustic events like phonemes occur in the frame of 10 mS to 100 mS [11]. Therefore, every speech signal is digitized into frames where duration of each frame is 23 mS for sampling frequencies 11025 Hz and that of 16 mS for the sampling frequency 8000 Hz.
- *Feature extraction* is the process by which the speech signal is converted to some type of parametric representation for further analysis and processing. This is a very important process in the high performance of CISI system. Appropriate features should be extracted from speech, otherwise the identification rate is influenced significantly. LPC and MFCC based feature vectors are extracted from the speech of each registered speaker.
- *Acoustic model.* It is not possible to use all extracted feature vectors to determine the identity of that speaker. Therefore, two modeling techniques: HMM and VQ are used to construct the acoustic model of each registered speaker. Then, for the identification of unknown registered speaker, its feature vectors are compared with each speaker's acoustic model present in the speaker database.

- *Pattern matching.* Then, for the identification of unknown registered speaker, its feature vectors are compared with each speaker's acoustic model present in the speaker database.
- *Identification decision.* When feature vectors of an unknown speaker are compared with acoustic model of each registered speaker, decision is made by computing distortion in the case of VQ modeling technique. Speaker's acoustic model having minimum distortion with unknown speech signal is recognized as a true speaker. For HMM, decisions are made by using maximum likelihood criteria.

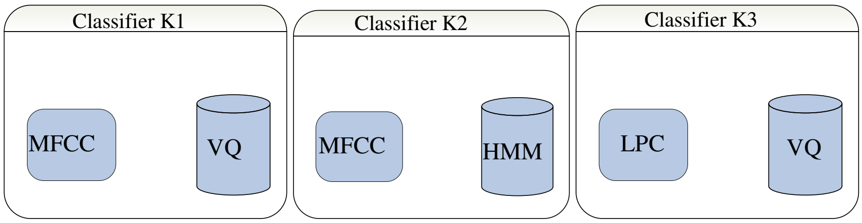


Fig. 1. Three Different Classifiers

In order to construct and check their performance of three individual classifiers K1, K2, and K3, we make following experimentation. K1 is obtained by combining MFCC with VQ, K2 and K3 by mixing respectively MFCC with HMM and LPC with VQ (see Figure 1). Additionally to make the system robust against noise, a consistent noise is added during the recording of each sentence. A database, having more than 700 voice samples, is recoded into two different sessions with the gap of two to three weeks to evaluate the performance of the system. It contains utterances of 44 speakers including 30 males and 14 females. Each speaker has recorded 6 different sentences at sampling frequencies 8000 Hz and 11025 Hz by using PRAAT software (http://www.fon.hum.uva.nl/praat/download_win.html). The list of these sentences is:

Sentence 1: Decimal digits from Zero to Nine
Sentence 2: All people smile in the same language.
Sentence 3: Betty bought bitter butter. But the butter was so bitter that she bought new butter to make the bitter butter better.
Sentence 4: Speakers recorded a random text from selected topic for 35 sec.
Sentence 5: Speakers recorded his/her roll number or employee ID.
Sentence 6: Speakers recorded a random text from selected topic for 12 sec.

Identification rates of all three classifiers at both sampling frequency 8000 Hz and 11025 Hz are respectively depicted in Figure 2 and Figure 3 when they are trained and tested by using the following sentences:

Training Sentence: Betty bought bitter butter. But the butter was so bitter that she bought new butter to make the bitter butter better.
Testing Sentence: All people smile in the same language.

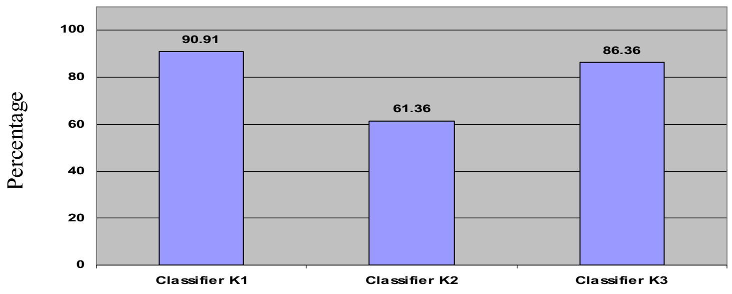


Fig. 2. Identification rate of classifiers at 8000 HZ

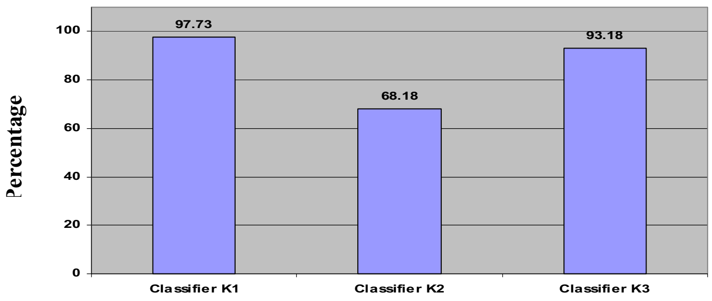


Fig. 3. Identification rate of classifiers 11025 HZ

Identification rate of these classifiers is computed by using the following relation:

$$\text{Identification Rate} = \frac{\text{Truly Identified Speaker}}{\text{Total No. of Speakers}}$$

Above experiments show that higher sampling frequency demonstrate good identification rate than lower sampling frequency, and classifier K1 (MFCC based VQ classifier) is better than all other classifiers at both sampling frequencies. Now, we use the results of sampling frequency 8000 HZ to construct Multiple Classifier System (MCS) because in that case even best classifier has identification of 90.91%. So, there is a lot room for the improvement of identification rate.

3 Multiple Classifier Speaker Identification System (MCSIS)

MCSIS uses the output of all three classifiers to make the joint decision about the identity of the speaker. The MCSIS can be categorized according to the level of classifier outputs which they use during the combination [12], [13]. There are three different levels of classifier's outputs:

- **Abstract Level.** Each classifier outputs the identity of a speaker only, and this level contains the lowest information.
- **Rank Level.** Classifiers provide a set of speaker identities ranked in order of descending likelihood.

- **Measurement Level.** It conveys the greatest amount of information about each particular speaker that may be correct one or not.

Different combination techniques [14]: Sum Rule, Product Rule, Min Rule, and Max Rule are used in this work to combine the normalized measurement level outputs of classifiers for the joint decision about the identity of the speaker.

3.1 Normalized Measurement Level Output of Classifiers

Measurement Level provides quite useful information about every speaker as shown in Table 1 and Table 2. First row of each table presents the measurement level output of Classifier K1 and K2 respectively, for speaker S1 when it is compared with 8 other speakers. A major problem within measurement level combination is the incomparability of classifier outputs [15]. As we can observe, the 1st row of both tables, it is clear that output of classifiers having different feature vectors differ in range and the outputs are incomparable. In consequence, before combining the outputs, it is necessary to treat these outputs of each classifier. Therefore, measurement level output of each classifier is normalized to the probabilities by dividing each element of the row by the sum of all the elements of that row.

Table 1. Measurement level Output for a Classifier using MFCC Features

	S 1	S 2	S 3	S 4	S 5	S 6	S 7	S 8
S 1	1.549	1.922	2.096	2.216	2.098	2.377	2.192	2.308
Normalized	0.092	0.115	0.125	0.132	0.125	0.142	0.131	0.138

Table 2. Measurement level Output for a Classifier using LPC Features

	S 1	S 2	S 3	S 4	S 5	S 6	S 7	S 8
S 1	0.585	0.692	1.429	0.742	0.754	1.310	1.395	1.694
Normalized	0.069	0.082	0.170	0.088	0.090	0.134	0.166	0.201

After normalization, the outputs of all classifiers lie in the interval of [0,1]. Now, we find a suitable combination technique for MCSIS. These techniques are discussed in following sections, which provide the better identification rate than individual classifiers.

3.2 Sum Rule (Linear Combination)

Linear combination is the simplest technique for MCS. For each speaker, sum of outputs of all classifiers is calculated. The decision of the true speaker depends on the maximum value obtained, after combination [13] [16] [17]. Suppose that there are 3 classifiers K1, K2, and K3, and five speakers (S1, S2, S3, S4, and S5). Outputs of these classifiers are represented by O1, O2, and O3. These output vectors are given as:

$$O1=[a_1\ \ a_2\ \ a_3\ \ a_4\ \ a_5]^T\ ,\ O2=[b_1\ \ b_2\ \ b_3\ \ b_4\ \ b_5]^T\ ,\ O3=[c_1\ \ c_2\ \ c_3\ \ c_4\ \ c_5]^T$$

a_j, b_j, c_j where $j = 1,2,3,4,5$ are positive real numbers. These output vectors are combined to make an output matrix which is

$$O = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \\ a_4 & b_4 & c_4 \\ a_5 & b_5 & c_5 \end{bmatrix}$$

The Sum rule is defined as

$$O_{sum} = \sum_{i=1}^{k} O_i \quad where \;\; i = 1,2,\cdots k$$

where O_i is the i^{th} column of output matrix. After combing by sum rule output matrix becomes

$$O_{sum} = \begin{bmatrix} a_1 + b_1 + c_1 \\ a_2 + b_2 + c_2 \\ a_3 + b_3 + c_3 \\ a_4 + b_4 + c_4 \\ a_5 + b_5 + c_5 \end{bmatrix}$$

When the value in the first row is larger than other values, the result of MCS is speaker1 (S1). Similarly, if value in the second row is larger than other values then the result of MCS is speaker2 (S2) and so on.

3.3 Product Rule (Logarithmic Combination)

Product rule, also called logarithmic combination, is another simple rule for classifier combination system. It works in the same manner than linear combination but instead of sum, the outputs for each speaker from all classifiers are multiplied [12], [16]. The product rule is defined as

$$O_{prod} = \prod_{i=1}^{k} O_i \quad where \;\; i = 1,2,3,...,k$$

When the output of any classifier for a particular speaker is zero, this value is replaced by a very small positive real number. After combining the output vectors of all classifiers, output matrix becomes:

$$O_{prod} = \begin{bmatrix} a_1 \cdot b_1 \cdot c_1 \\ a_2 \cdot b_2 \cdot c_2 \\ a_3 \cdot b_3 \cdot c_3 \\ a_4 \cdot b_4 \cdot c_4 \\ a_5 \cdot b_5 \cdot c_5 \end{bmatrix}$$

The decision criterion is similar as the Sum rule.

3.4 Min Rule

Min rule combination method measures the likelihood of a given speaker by finding the minimum normalized measurement level output for each speaker. Then final decision for identifying a speaker is made by determining the maximum value [12] [13].

Example 1: Consider an output matrix which is obtained by combining the output vectors of the three classifiers. Each column of the matrix represents the output of a classifier for five speakers.

$$O = \begin{bmatrix} 0.0 & 0.3 & 0.2 \\ 0.4 & 0.3 & 0.2 \\ 0.6 & 0.5 & 0.4 \\ 0.0 & 0.0 & 0.1 \\ 0.2 & 0.1 & 0.3 \end{bmatrix}$$

Each element of O_{min} is the minimum value selected from each row of the output matrix O. Each row corresponds to the output of a classifier for a particular speaker. The final decision is the maximum value of the vector O_{min} which is 0.4. This value shows that the true speaker is the speaker number 3.

$$O_{\min} = [0.0 \quad 0.2 \quad 0.4 \quad 0.0 \quad 0.1]^T$$

3.5 Max Rule

In the Max rule, the combined output of a class is the maximum value of the output values provided by different classifiers for the corresponding speaker [12] [16]. For a better explanation, consider the following example.

Example 2: Assume that we have three classifiers and five speakers. Their output matrix is given below:

$$O = \begin{bmatrix} 0.0 & 0.3 & 0.2 \\ 0.4 & 0.3 & 0.2 \\ 0.6 & 0.5 & 0.4 \\ 0.0 & 0.0 & 0.1 \\ 0.2 & 0.1 & 0.3 \end{bmatrix}$$

The combined output vector is obtained by selecting maximum value from each row of the output matrix. The resultant vector is

$$O_{\max} = [0.3 \quad 0.4 \quad 0.6 \quad 0.1 \quad 0.3]^T$$

Maximum value in the vector O_{max} is 0.6 which corresponds to speaker number 3. So, the joint decision of all the classifiers is the speaker3.

3.6 Confusion Matrix

Confusion matrix is a handy tool to evaluate the performance of a classifier. It contains the information of both truly identified speakers as well as misclassified speakers [15] [18]. Each column of this matrix represents the true speaker. Let us assume that 50 voice samples of speaker3 are tested by the identification system. If all these voice samples are truly identified then value at the 3rd row, 3rd column will be 50 with zeros elsewhere. On the other hand, if values of 1st , 2nd , 3rd , 4th , and 5th row of 4th column are 3, 1, 0, 41, and 5 respectively show that 3 times speaker4 is misclassified as speaker1, 1 time speaker4 is misclassified as speaker2, 45 times speaker4 is truly identified, and 5 times speaker4 is misclassified as speaker5 by the system. A confusion matrix is shown in Figure 4.

$$\begin{array}{cc|ccccc} & & \multicolumn{5}{c}{\text{True Speakers}} \\ & & 1 & 2 & 3 & 4 & 5 \\ \hline & 1 & 50 & 0 & 0 & 3 & 0 \\ & 2 & 0 & 47 & 0 & 1 & 0 \\ \text{Identified Speakers} & 3 & 0 & 2 & 50 & 0 & 1 \\ & 4 & 0 & 0 & 0 & 41 & 0 \\ & 5 & 0 & 1 & 0 & 5 & 49 \end{array}$$

Fig. 4. A Confusion Matrix

4 Results

The identification rates of MSCIS, after applying Sum, Product, Min, and Max Rule on the output of individual classifiers, are presented in Figure 5. Max Rule as a Combination rule in MCS has shown an increase of 4.54% in identification rate than that of best individual classifier which was 90.91%.

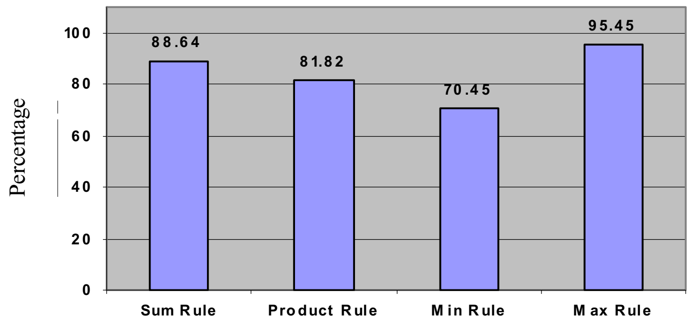


Fig. 5. Identification Rates of Combination Techniques

Some combination techniques show poor identification rate even than individual classifiers. A comparison between identification rates of best individual classifier K1, best combination technique (Max Rule), and confusion matrix is depicted in Figure 6.

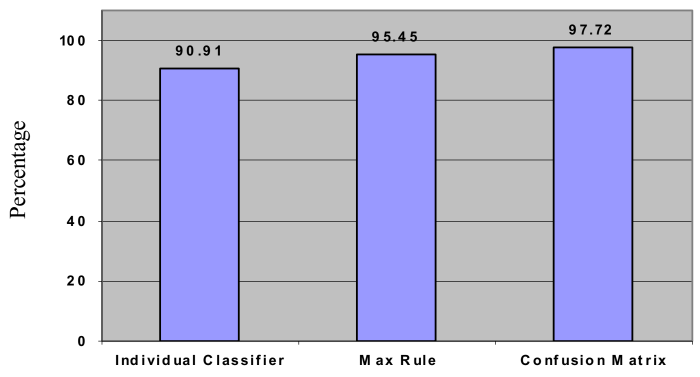


Fig. 6. Comparison of Confusion Matrix Technique with Max Rule and Individual Classifier

5 Conclusion and Future Work

MFCC based VQ classifier, LPC based VQ classifier, and MFCC based HMM are combined to make Multiple Classifier System (MCS). Normalized measurement level outputs of the classifiers are combined by using Min Rule, Max Rule, Product Rule and Sum Rule. Combination technique Max Rule demonstrated good results as compared to other combination technique. Max rule improves identification rate by 4.54% than best individual classifiers. But when classifiers are combined by using Confusion matrix, it shows improvement of 6.81% than best individual classifier and 2.27% than Max Rule in the proposed multiple classifier text-independent system. Experiment shows that Confusion matrix based MCS produces excellent result as compared to each individual classifier. These results are also better than various combination techniques, i.e. Sum, Product, Min Rule, and Max Rule.

In the identity of the speaker case studied, our proposed MCS for CISI system gives the same importance to the results obtained by each classifier. In order to enhance the performance in the decision process, the output of a classifier can be pondered by a weight, when its performance is better than other classifiers within the environment tested. It is our future validation in which we continuous making tests.

References

[1] Furui, S.: Recent Advances in Speaker Recognition. Pattern Recognition Letter 8(9), 859–872 (1997)

[2] Chen, K., Wang, L., Chi, H.: Methods of Combining Multiple Classifiers with Different Features and Their Application to Text-independent Speaker Identification. International Journal of Pattern Recognition and Artificial Intelligence 11(3), 417–445 (1997)

[3] Reynolds, D.A.: An Overview of Automatic Speaker Recognition Technology. Proc. IEEE 4, 4072–4075 (2002)

[4] Godino-Llorente, J.I., Gómez-Vilda, P., Sáenz-Lechón, N., Velasco, M.B., Cruz-Roldán, F., Ballester, M.A.F.: Discriminative Methods for the Detection of Voice Disorder. In: A ISCA Tutorial and Research Workshop on Non-Linear Speech Processing, The COST-277 Workshop (2005)

[5] Xugang, L., Jianwu, D.: An investigation of Dependencies between Frequency Components ans Speaker Characteristics for Text-independent Speaker Identification. Speech Communication 2007 50(4), 312–322 (2007)

[6] Huang, X.D., Ariki, Y., Jack, M.A.: Hidden Markov Model for Speech Recognition. Edinburgh University Press, Edinburgh (1990)

[7] Linde, Y., Buzo, A., Gray, R.M.: An Algorithm for Vector Quantizer Design. IEEE Transaction on Communications 28, 84–95 (1980)

[8] Higgins, J.E., Damper, R.I., Harris, C.J.: A Multi-Spectral Data Fusion Approach to Speaker Recognition. In: Fusion 1999, 2nd International Conference on Information Fusion, Sunnyvale, CA, pp. 1136–1143 (1999)

[9] Premakanthan, P., Mikhael, W.B.: Speaker Verification /Recognition and the Importance of Selective Feature Extraction:Review. In: Proc. of 44th IEEE MWSCAS 2001, vol. 1, pp. 57–61 (2001)

[10] Razak, Z., Ibrahim, N.J., Idna Idris, M.Y., et al.: Quranic Verse Recitation Recognition Module for Support in J-QAF Learning: A Review. International Journal of Computer Science and Network Security (IJCSNS) 8(8), 207–216 (2008)

[11] Becchetti, C., Ricotti, L.P.: Speech Recognition Theory and C++ Implementation. John Wiley & Sons, Chichester (1999)

[12] Kittler, J., Hatef, M., Duin, R.P.W., Mates, J.: On Combining Classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(3), 226–239 (1998)

[13] Kuncheva, L.I., Bezdek, J.C., Duin, R.P.W.: Decision Templates for Multiple Classifier Fusion: An Experimental Comparison. Pattern Recognition 34(2), 299–314 (2001)

[14] Shakhnarovivh, G., Darrel, T.: On Probabilistic Combination of face and Gait Cues for Identification. In: Proc. 5th IEEE Int'l Conf. Automatic Face Gesture Recognition, pp. 169–174 (2002)

[15] Ho, T.K., Hull, J.J., Srihari, S.N.: Decision Combination in Multiple Classifier Systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 16(12), 66–75 (1994)

[16] Tumer, K., Ghosh, J.: Linear and Order Statistics Combiners for Pattern Classification. In: Sharkey, A. (ed.) Combining Artificial Neural Networks, pp. 127–162. Springer, Heidelberg (1999)

[17] Chen, K., Chi, H.: A Method of Combining Multiple Probabilistic Classifiers through Soft Competition on Different Feature Sets. Neuro Computing 20(1-3), 227–252 (1998)

[18] Kuncheva, L.I., Jain, L.C.: Designing Classifier Fusion systems by Genetic Algorithms. IEEE Tran. on Evolutionary Computation 4(4), 327–336 (2000)

Towards One-Class Pattern Recognition in Brain Activity via Neural Networks

Omer Boehm[1], David R. Hardoon[2], and Larry M. Manevitz[1]

[1] University of Haifa
Computer Science Department
Haifa, Israel 31905
manevitz@cs.haifa.ac.il, oboehm@cs.haifa.ac.il
[2] Institue for Infocomm Research
Machine Learning Group
A*Star, Singapore
drhardoon@i2r.a-star.edu.sg

Abstract. In this paper, we demonstrate how one-class recognition of cognitive brain functions across multiple subjects can be performed at the 90% level of accuracy via an appropriate choices of features which can be chosen automatically. The importance of this work is that while one-class is often the appropriate classification setting for identifying cognitive brain functions, most work in the literature has focused on two-class methods.

Our work extends one-class work by [1], where such classification was first shown to be possible in principle albeit with an accuracy of about 60%. The results are also comparable to work of various groups around the world e.g.[2], [3] and [4] which have concentrated on two-class classification.

The strengthening in the feature selection was accomplished by the use of a genetic algorithm run inside the context of a wrapper approach around a compression neural network for the basic one-class identification. In addition, versions of one-class SVM due to [5] and [6] were investigated.

Keywords: One-class classification, fmri, fmri-classification, Neural networks, Genetic algorithms.

1 Introduction

In recent years, identifying cognitive activity from direct physiological data by using functional Magnetic Resonance Imaging (fMRI) studies as data and identifying the cognitive activity directly from the brain scans has become a real possibility. (See [2,4,3,1,7], to name a few.) This correspondence between physiological information and specific cognition lies at the very heart of the goals of brain science.

Note that this work is, in a sense, the opposite of another area of central concern for brain science, specifically, the problem of identifying which areas of the brain are associated with various cognitive activity. However, there is a strong synergy between these two activities. While it might, in principle, be possible to identify the cognitive activity from full brain data, most researchers in this area, starting with [4,2] have realized that the strong noise to signal ratio in brain scans requires aggressive feature selection.

This noise to signal ratio has several origins:

– The inherent noise in the technological scan;
– The variability within a single subject;
– The fact that a brain is actually performing many tasks simultaneously and one can not control for all of them;
– Brains are physically distinct across individuals and the mappings between them are only approximate [8];
– MRI technology has limited resolution, so in a sense the original data is always "smeared" in the spatial dimension.
– Activity levels are measured indirectly via blood oxygenation, so the data is also "smeared" with respect to time.

In addition, considering the dimensionality of the data, one always has very few data points. A typical scan has about 120 thousand voxels with real values, while the expense and difficulty in acquiring fMRI data of an individual means that the complete data set is in the order of a hundred samples. Thus, the problem being tackled has small data size, large dimensionality, and a large noise to signal ratio. A priori it would seem an unlikely endeavor. Nonetheless, the results reported (beginning with Cox and Savoy [2] and with Mitchell et. al [4]) show that it is possible.

In these works, methods to aggressively reduce non-relevant (noise) features were applied. Note that if one manages to reduce the number of features, one is essentially finding the voxels of the brain that are associated with the cognitive problem; i.e. the complementary problem.

In this work we decided to focus on one-class classification rather than two-class classification, for reasons that will be discussed below. (In our opinion it is often the appropriate setting for this application). (See [9,6] for some further information on one-class approaches and [10,11] for other interesting applications of one-class methods.) The one-class classification here was used as an evaluator in two different search approaches. We used a "wrapper approach" [12] to find the relevant features with partial success. As a result, we decided to combine this with a genetic algorithm to automate and improve the search for features.

We were able to consistently find features that allow differential classification at about the 90% level which now makes this methodology applicable. (In contrast, results on this task without feature selection were about 60% which is similar to the reported results of [1] on a motor task.) However, as discussed below, for evaluation of the effectiveness of this method, we need to use test data from both classes. While this is necessary and standard for testing one-class methods, from one point of view, this contaminates the "one-class" philosophy because one has to perform such evaluation many times in the genetic algorithm during the feature selection. In future work, we hope to alleviate this problem by showing that the results are somewhat robust in the choice of the data in the second class.

As a secondary point, we expected to see that the selected features would be focused in specific and contiguous areas of the brain in visual cortex. (For example, "faces" features are expected to be in an area of the temporal lobe known as the fusiform gyrus [13]). Surprisingly, this was not the case. In fact, no voxels were found that were persistent between runs. Our interpretation is that, the information needed for classification has percolated and it suffices to only sample these dimensions, and the genetic algorithm picks out specific samples which can vary.

The paper is organized as follows : section 2 discusses one-class versus two-class classification; section 3 briefly describes the data set the experiments were performed on; section 4 discusses feature reduction and our manual search; sections 5 describes how we used the genetic algorithm to this task; Section 6 discusses issues related to the "converse problem" of finding areas associated with these tasks and finally, section 7 includes a summary and our intended future directions.

2 One-Class versus Two-Class Classification

The problem of classification is how to assign an object to one of a set of classes which are known beforehand. The classifier which should perform this classification operation (or which assigns to each input object an output label), is based on a set of example objects. This work focuses on the problem of one-class classification. In this case , an object should be classified as an object of the class or not. The one-class classification problem differs in one essential aspect from the conventional classification problem. In one-class classification it is assumed that only information of one of the classes, the target class, is available. This means that just example objects of the target class can be used and that no information about the other class of outlier objects is present during training. The boundary between the two classes has to be estimated from data of only the normal, genuine class. The task is to define a boundary around the target class, such that it accepts as much of the target objects as possible, while it minimizes the chance of accepting other objects.

When one is looking for a two-class (or n-class with $n \geq 2$) the assumption is that one has representative data for each of the classes and uses them to discover separating manifolds between the classes. While the most developed machine learning techniques address this case, this is actually a very unusual situation.

While one may have invested in obtaining reasonably representative data addressing one-class, it is unusual to have a representative sample of its complement in two-class learning. Similar problem can be exhibited in the information retrieval field e.g. querying some search engine for 'houses' will probably yeild reasonable results, but looking for anything other than a house i.e. search for 'not houses' would probably yeild poor results. The same is true for the n-class case.

A significant weakness of n-class filters is that they must be re-created as data for each class is obtained, and divisions between sub-classes must all be trained separately. Furthermore, essentially, one can never have sufficient data to distinguish between class A and "anything else". Thus, while one may initially have data representing class A, B and C, one must then use two-class methods to find a filter distinguishing between class A and B, class A and C, and class B and C; or alternatively one must find a filter between class A and class (B or C) and class B between (A or C); etc. two-class classification then becomes overly specific to the task at hand. The assumption in using these filters will be that the data comes from one of these classes. Should one wish to add class D, then existing filters must be retrained, and many additional filters distinguishing D from the rest of the above classes must be trained.

It is more natural to imagine a scenario where data is gathered for a particular kind of cognitive task and then, when data for another task is gathered, a different filter is made for the new class. Thus one can incrementally build up a library or "battery" of classification filters; and then test a new data point by this battery. Of course, it would then be possible for a data point to pass several such filters.

However, as expected, in earlier results by [1] the results for two-class classification were superior to those of one-class classification. Their work showed that while one-class classification can be done in principle, for this fMRI task, their classification results (about 60%) were not sufficient for an actual application.

In this work, we have remedied this problem, by showing that one can obtain, automatically, filters with accuracy close to their two-class cousins. The main methodology was finding the appropriate features. This was a reasonable hope given the large dimension of features given by the fMRI map (which were all used in [1]) and since, as described above, most of these features can be thought of as "noise" for this task.

To do this we proceeded with the following main tools:

1. A choice of a one-class classifier approach. The two that we considered were
   (a) The compression neural network [14,9].
   (b) Different versions of one-class SVM [6,15]
2. The use of the wrapper approach [12] to judge the quality of features.
3. A manual ternary search proceeding by a ternary dissection approach to the brain (at each stage using the one-class wrapper as an evaluator.)
4. Finally, the use of a genetic algorithm [16] to isolate the best features.

The one-class learning method was used to perform the evaluation function in the manual search and the genetic algorithm.

Each of these steps has its own complications and choices. For example: Step 1a requires choosing an appropriate compression ratio for the one-class neural network and, of course, choosing the training method. Step 1b has many variants;

we did not exhaust all of them, but we found the results too sensitive to the choices and so in the end used a version of 1a almost exclusively.

Step 3, being manual took too long; we used its results to help decide on the initial conditions of the genetic algorithm.

In both step 3 and step 4, there is a need to evaluate the quality of the features for discrimination. While it is standard in one-class to use the second class data to evaluate the classifier, in this case, the results of this evaluation implicitly affects the choice of features for the next step, and so distorts the pure one-class learning method.

We have chosen to ignore this problem in this work; partially due to lack of time and partially because the results seem robust to the choice of the second class data. Below, in future work, we sketch how we hope to eliminate this problem.

3 Task and Data Description

In the experiment that provided the data analyzed here, four subjects, inside a MRI-scanner, were passively watching images belonging to five different semantic categories as follows: human faces, houses, patterns, objects, blank image. The blank image is considered as 'null', as if nothing is viewed. Normalization between individuals were carried as suggested in [8] [17].

The time-course reference of the experiment is built from each subject viewing a sequence of the first four categories separated by the "blank" category i.e. blank, face, blank, house, blank, pattern, blank, object, blank. 147 fMRI scans are taken over this sequence per subject; thus the 'raw' data consists of 21 data points for the first four semantic categories and 63 data points for the blank image.

The individual fMRI images are dicom format (58 image slices) of size 46x46 overall consisting of 122,728 real-valued voxels.

4 Feature Reduction and Manual Search

4.1 Results without Feature Reduction

In some preliminary work, we ran this task without feature reductions, but because of computational limitations at the time, we used every 5th slice out of the 58 available. Thus the data was represented by 13,800 features. The one-class task was run both with a compression neural network (60% compression) and with a version of one-class SVM on the cross individual data. In this experiments we used 38 positive samples for training and 25 positive and 25 negative samples for testing repeated for 10 random runs. Table 1 shows the success rate when trained on each category vs. blank for the neural network approach while Table 2 shows the results for one class SVM.

We see that we were unable to produce results above random using the one-class SVM methodology. On the other hand, the compression neural network

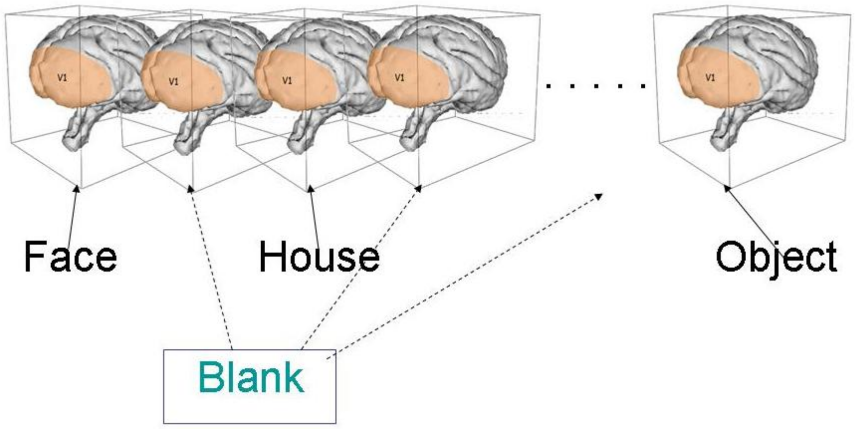


Fig. 1. Illustration of the fMRI scans taken during the experiment

Table 1. Combined Individuals - Bottleneck neural network with 60% compression

	Face	Pattern	House	Object
Blank	56.6% ± 3.8%	58% ± 3.7%	56.2% ± 3.1%	58.4% ± 3.1%

Table 2. Combined Individuals - One-class SVM Parameters Set by Subject *A*

	Face	Pattern	House	Object
Blank	51.4% ± 2.55%	52.20% ± 3.49%	53.7% ± 3.77%	52.4% ± 2.9%

produced significant results but only in the 60% level. Tests for trained category versus other categories were similar.

This is comparable to results reported in [1] on identifying the fMRI correlate of a motor task (”finger flexing”) using one-class learning (about 59% obtained using either a compression neural network or a one-class SVM).

4.2 Feature Reduction via Manual Search

To recapitulate, our approach reduces the question of finding the features, to a search amongst the subsets in the space of features. In this work, we have examined both one-class SVM and compression neural networks as the machine learning tool. These were investigated in [1] where it was found that the neural network approach worked somewhat better. This is not so surprising when considering the work of [15], where it was shown, in a comparative study in a textual classification task, that while both seem to have similar capabilities; the SVM was much more sensitive to the choice of parameters.

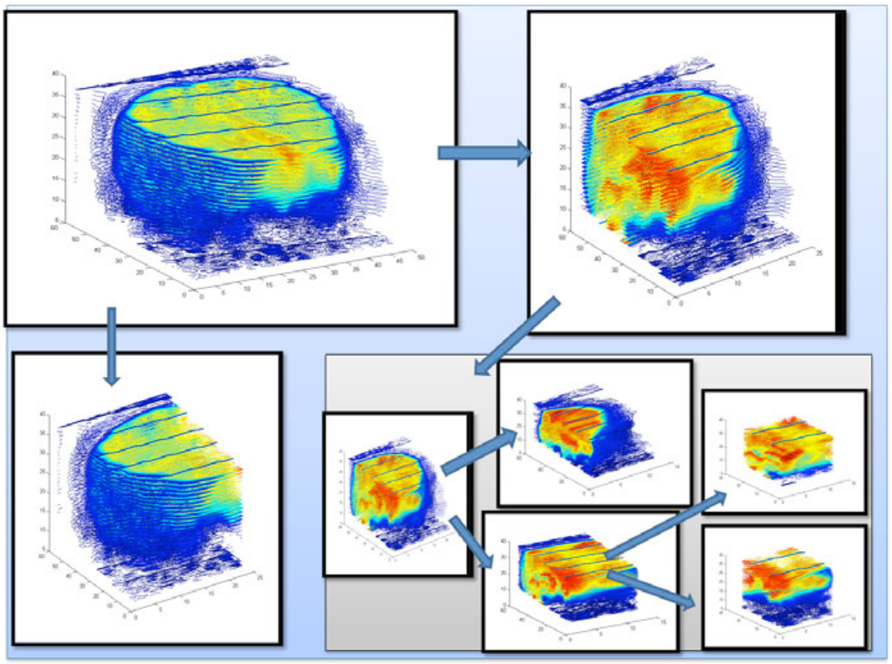


Fig. 2. Conceptual manual binary search via the one-class bottleneck neural network

The main emphasis of this work is the feature selection, using the wrapper approach and the genetic algorithm approach. We followed two paths: initially we worked by hand and did a primitive, greedy search on the subsets as follows:

- First, start with a "reasonable" area of the data scan; i.e. all background dead area cropped out; the most external levels of the brain discarded. That is, the raw scan had about 120,000 real valued voxels; after reduction we had about 70,000 voxels.
- Second, divide (various options to do that) the brain into overlapping two or three geometrically contiguous boxes (by selecting along one dimension) - run the classifier and discard the lowest returns; Continue with the best box as long as it classifies better than the previous loop.
- When all boxes do worse; consider either (i) perform a different division of the boxes along the same dimension as before, but now of different sizes that overlaps the previous chosen boxes or (ii) select the boxes by slicing in a different dimension. (i.e. if the search was on boxes defined by the row indices, now use the best row indices found and try to create boxes with different column indices).
- Cease when no improvement is found.

Table 3. Manual ternary search via the one-class bottleneck neural network for 'faces' data. * indicates the chosen part. (If no part is chosen, the current path is terminated, and a different division step is performed. See text).

Iteration	[rows, columns, height]	# features	Houses	Objects	Patterns	Blank	Avg
1	[1-17,1-39,1-38]	25194	58%	56%	55%	60%	57%
	[15-33,1-39,1-38] *	28158	62%	55%	64%	65%	62%
	[30-48,1-39,1-38]	28158	55%	52%	50%	60%	54%
2	[15-33,1-39,1-15]	11115	61%	63%	55%	60%	60%
	[15-33,1-39,13-30] *	13338	69%	68%	72%	70%	70%
	[15-33,1-39,27-38]	8892	58%	57%	60%	60%	59%
3	[15-23,1-39,13-30]	6318	63%	69%	68%	62%	66%
	[20-26,1-39,13-30] *	4914	70%	67%	**76%**	79%	73%
	[25-33,1-39,13-30]	6318	60%	67%	70%	75%	68%
4	[20-23,1-39,13-30] *	2808	**74%**	70%	71%	73%	72%
	[22-25,1-39,13-30]	2808	65%	**73%**	60%	**80%**	71%
	[24-26,1-39,13-30]	2106	70%	69%	69%	68%	69%
5	[20-21,1-39,13-30]	1404	67%	65%	74%	63%	67%
	[21-22,1-39,13-30]	1404	60%	63%	70%	64%	64%
	[22-23,1-39,13-30]	1404	65%	63%	72%	68%	67%
6	[20-23,1-18,13-30]	1296	67%	66%	70%	72%	69%
	[20-23,19-39,13-30]	1512	67%	70%	72%	78%	72%

Figure 2 illustrates a sample process of the manual greedy binary search. The assumption was that the task 'relevant' features reside in a relatively small contiguous chunk in the brain.

Following this work, we were able to produce the following Table 3 of results: (obtained on one of the possible search paths) Each data point (3D matrix) which originally contained about 120,000 features was reduced as explained above into about 70,000 features .($[58 \times 46 \times 46] \rightarrow [48 \times 39 \times 38]$).

Table 3 represents the results in a specific run for the 'faces' (fMRI data acquired for subjects while viewing images of faces). We used a bottleneck neural network, with compression rate of 60%, which was trained solely for the 'faces' data and then tested against the rest of the categories. This was averaged over 5-folds. The decision how to continue was according to the average over all categories. As can be seen, this method brought us up to 80% accuracy on blank data, and 72% on average.

For a control and comparison, we also considered random selection of about the same proportion of features; and the results were not much above random.

5 Feature Reduction and the Genetic Algorithm

It is clear that this way of work is very tedious and there are many possible intuitive choices. In an attempt to automate it, we decided, to apply a genetic algorithm [16] approach to this problem, although the computational requirements became almost overwhelming.

During experimentations we implemented and tested a variety of configurations for the genetic algorithm. In general, each gene representation serves as a "mask" where a "1" indicates that a feature is chosen.

We used population sizes in the range of 30 to 50. In the initial generation, the creation function typically set "1" for about 10% of the features selected randomly.

A typical configuration included

– the genome representation e.g. bit strings, three dimensional matrices. (the matrix dimensions were set to be same as the "trimmed" data points).
– a selection function e.g. stochastic uniform, remainder, uniform and roulette.
– a reproduction method e.g. considered different amounts of elite members, different crossover fractions and various crossover options i.e. two-point crossover for bit string representation, or two planes crossing a cube for three dimensional matrix representation.
– a mutation function e.g. Gaussian, uniform, adaptive feasible etc.

The evaluation methods are the heart of the genetic algorithm. Each implementation included similar steps, i.e. similar pseudo code, and the differences were in the classifier type and data manipulations due to the different representations. The evaluation method works as follows: Given a gene, recreate the data by masking the gene (mask) over each one of the data points. The newly created data set after this action is a projection of the original data set and should have a significantly smaller dimension in each generation, due to the genetic pressure resulting from choosing precise features for classification. This smaller dimension also results in much faster classifier runs. Divide the new data into three parts :

– training data (60%) - taken from one class.
– threshold selection and testing data (20%) - taken from two classes.

Train the one-class classifier (either a bottleneck Neural Network or one-class SVM) over the training data (of all subjects)

Use threshold selection dedicated data and the trained classifier, to determine the best separating threshold.

Finally test using the remaining testing dedicated data and calcluate a success rate. The final evaluation function of the gene uses a weighted average of the

Table 4. Genetic algorithm results. The genetic algorithm was able to find a filter for each class with a success rate almost similar to the ones produced for the two-class classifiers.

	Faces	Houses	Objects	Patterns
Faces	-	84%	84%	92%
Houses	84%	-	83%	92%
Objects	83%	91%	-	92%
Patterns	92%	85%	92%	-
Blank	91%	92%	92%	93%

Table 5. Comparison between one-class results without feature reduction and with feature reduction via the genetic algorithm between trained classes and blank

	Faces	Houses	Objects	Patterns
Neural network without Feature Reduction	57%	58%	56%	58%
Neural network with Genetic Feature Reduction	91%	92%	92%	93%

success rate i.e. the number of the data points which were correctly classified and the variance of each testing error from the threshold. That is, the evaluation function tries to take into account the level of the certainty of each test answer.

We produced the results in Table 4 after 100 generations. During the run, we kept track of 'good' genes whose evaluation rate exceeded the weighted average 80, and then used the best ones.

In Table 5, we reproduce the results from Table 1 and the corresponding row of Table 4. The dramatic increase in accuracy is evident. Similar increases can be seen in all the other rows of Table 4.

6 Location of Areas of Brain Associated with Cognitive Tasks

Having discovered features appropriate for classification; it is interesting to enquire whether or not these features are local, i.e. presented in a particular area of the brain, or distributed. Of course, this can only be asked up to the resolution of the fMRI data themselves.

To get a feel for this, we used Matlab visualization tools. We can show in figure 3 a three dimensional location of the features (of one of the best genes

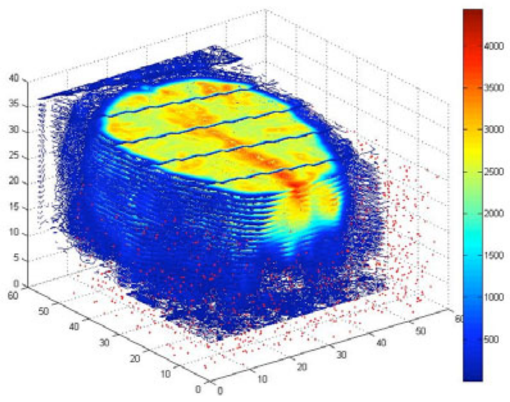


Fig. 3. A visualization of a 'face' data point and a chromosone (set of features) which was able to show 91% separation success rate. The red dots indicate selected features.

found by the genetic algorithm) unified with a high resolution contour brain slices.

Surprisingly, although we have not yet quantitatively analyzed all of these results, a visual analysis does not indicate, contrary to expectations, a strong locality of the features. Thus we can not at this stage state which areas of the brain are important for the classification of each task. It is not inconsistent with our results that the best feature selection requires a non-trivial combination of areas. Another possibility, as mentioned above, is that areas of importance in the cortex need only be sampled to provide sufficient classification information and the genetic algorithm just converges in each run to a different such sample. A clarification of this issue awaits further analysis.

7 Summary and Future Work

Recognizing cognitive activities from brain activation data is a central concern of brain science. The nature of available data makes this application, in the long term, a one-class activity; but until now only two-class methods have had any substantial success. This paper successfully solves this problem in the sample visual task experimented on.

– We have shown that classifying visual cognitive tasks can be done by one-class training techniques to a high level of generalization.
– We have shown that genetic algorithms; together with the one-class neural network compression network can be used to find appropriate features that, on the one hand, increase the accuracy of the classification to close to that obtainable from two-class methods.
– Preliminary results show that this method may indicate non compact areas of the brain must cooperate in order to be critically associated with the cognitive task

This work needs to be extended to other (non-visual) cognitive tasks; and it needs to be seen to what resolution the work can be carried out. Can specific styles or specific faces of people be identified from these kind of mechanisms? Is this a theoretical limit on either the accuracy or the resolution?

References

1. Hardoon, D.R., Manevitz, L.M.: fmri analysis via one-class machine learning techniques. In: Proceedings of the Nineteenth IJCAI, pp. 1604–1605 (2005)
2. Cox, D., Savoy, R.: Functional magnetic resonance imaging (fmri) "brain reading": detecting and classifying distributed patterns of fmri activity in human visual cortex. NeuroImage 19, 261–270 (2003)
3. Mourao-Miranda, J., Reynaud, E., McGlone, F., Calvert, G., Brammer, M.: The impact of temporal compression and space selection on svm analysis of single-subject and multi-subject fmri data. NeuroImage (2006), doi:10.1016/j.neuroimage.2006.08.016

4. Mitchell, T.M., Hutchison, R., Niculescu, R.S., Pereira, F., Wang, X., Just, M., Newman, S.: Learning to decode cognitive states from brain images. Machine Learning 57, 145–175 (2004)
5. Scholkopf, B., Platt, J., Shawe-Taylor, J., Smola, A., Williamson, R.: Estimating the support of a high-dimensional distribution. Technical Report MSR-TR-99-87, Microsoft Research (1999)
6. Manevitz, L., Yousef, M.: One-class svms for document classification. Journal of Machine Learning Research 2, 139–154 (2001)
7. Carlson, T.A., Schrater, P., He, S.: Patterns of activity in the categorical representations of objects. Journal of Cognitive Neuroscience 15(5), 704–717 (2004)
8. Talarich, J., Tournoux, P.: Coplanar stereotaxic atlas of the human brain. Thieme Medical, 122 (1988)
9. Japkowicz, N., Myers, C., Gluck, M.A.: A novelty detection approach to classification. In: International Joint Conference on Artificial Intelligence, pp. 518–523 (1995)
10. Sato, J., da Graca Morais Martin, M., Fujita, A., Mourao-Miranda, J., Brammer, M., Amaro Jr., E.: An fmri normative database for connectivity networks using one-class support vector machines. Human Brain Mapping, 1068–1076 (2009)
11. Yang, J., Zhong, N., Liang, P., Wang, J., Yao, Y., Lu, S.: Brain activation detection by neighborhood one-class svm. Cognitive Systems Research (2008) (in press, corrected proof)
12. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence (1997)
13. Kanwisher, N., McDermott, J., Chun, M.M.: The Fusiform Face Area: A Module in Human Extrastriate Cortex Specialized for Face Perception. J. Neurosci. 17, 4302–4311 (1997)
14. Cottrell, G.W., Munro, P., Zipser, D.: Image compression by back propagation: an example of extensional programming. Advances in Cognitive Science 3 (1988)
15. Manevitz, L., Yousef, M.: Document classification via neural networks trained exclusively with positive examples. Neurocomputing 70, 1466–1481 (2007)
16. Goldberg, D.E.: Genetic Algorithms in search, Optimization & Machine learning. Addison-Wesley Publishing Company, Inc., Reading (1989)
17. Hasson, U., Harel, M., Levy, I., Malach, R.: Large-scale mirror-symmetry organization of human occipito-temporal objects areas. Neuron 37, 1027–1041 (2003)

Real Time Tracking of Musical Performances

Antonio Camarena-Ibarrola[⋆] and Edgar Chávez

Universidad Michoacana de San Nicolás de Hidalgo
Morelia, Mich., México
{camarena,elchavez}@umich.mx

Abstract. Real time tracking of musical performances allows for implementation of virtual teachers of musical instruments, automatic accompanying of musicians or singers, and automatic adding of special effects in live presentations.

State of the art approaches make a local alignment of the score (the target audio) and a musical performance, such procedure induce cumulative error since it assumes the rendition to be well tracked up to the current time. We propose searching for the k-nearest neighbors of the current audio segment among all audio segments of the score then use some heuristics to decide the current tracked position of the performance inside the score.

We tested the method with 62 songs, some pop music but mostly classical. For each song we have two performances, we use one of them as the score and the other one as the music to be tracked with excellent results.

Keywords: entropy, index, proximity.

1 Motivation

Real time tracking of musical performances consists in establishing the position of the current short segment of audio of a musical rendition of a test song as is being played in relation to the score (the target audio). Among the possible applications of real time tracking of musical performances we will briefly explain three of them: The virtual music teacher, Automatic accompanying of single players or singers, and automatic adding of special effects in live presentations.

For the virtual music teacher, the score is a well played musical piece interpreted by a trained musician. The music to be tracked is that generated by a student. Normally, a music teacher handles only one student at a time, as the student plays his instrument, the teacher gives him correction indications or gives his approval at the end. A virtual teacher should do the same, it must use the audio signal generated by the student, process it in real time and provide indications to the student accordingly. Of course, the great advantage of a virtual teacher is that it may multiply easily to manage many students simultaneously.

[⋆] Corresponding author.

For automatic accompanying of a singer or a single player of an instrument, the score is an audio signal generated by the single player or singer without accompaniment (e.g. without orchestra) and the music to be tracked is a live performance of the same musician. The system plays the accompanying music and at the same time generates actions (e.g. To introduce delays in the background music). This application can be thought of as an intelligent Karaoke.

For the automatic adding of special effects application, the score is the audio signal recorded during a session of practice of the event, the tracked audio signal is captured in the live event. Examples of actions are turning lights on or off, playing sound effects or even launching fireworks at specific instants marked in the score.

2 Related Work

Real time tracking music systems are characterized by two aspects; The features extracted from the audio signal, and the technique used to align the score with the performance. Features are suppose to have every piece of audio in both the score and the performance characterized. In [1] the dynamic beat period of the signal is determined. In [2] the energy, delta energy, zero crossings, fundamental frequency and delta fundamental frequency are the relevant features extracted from the signal for audio tracking purposes. In [3,4], the global pitch, chroma values, cepstral flux and the spectrum were the collection of features chosen for tracking performances on line.

Once the signal's feature set has been defined, an alignment technique is normally used to relate each time instant within the musical performance to the corresponding time in the score. The classic approach to align two sequences is known as *Dynamic Time Warping* (DTW) [5]. DTW consists of finding the optimal warping function which states how one of the sequences should be shortened or stretched to reduce the differences between both sequences down to a minimum. DTW was originally designed to align two sequences that are both known *a-priori*. For our purpose (tracking musical performances in real time) only one of the sequences is known *a-priori* (the score), the alignment has to be performed as the other sequence arrives (the performance). Recently, in [6] and [7] a variation of DTW, an "on-line time warping" was proposed as an adaptation to allow for tracking musical performances on-line, the onsets of tones and the increases in energy in frequency bins were used as features. The on-line time warping algorithm attempts to predict the optimal warping path by partially filling the matrix of costs with windows of a fixed size (such size is a free parameter of the algorithm), if the minimum cost is found in the rightmost column of the window, then the filling advances column-wise, if it is found in the downmost row of the window, then the filling of the costs' matrix advances row-wise. The algorithm advance both in row and column when the minimum cost is found in the pseudo-diagonal. The real diagonal is not really known since one of the series length (the musical performance under tracking) is unknown. The "on-line time warping" algorithm toggles from row to column and viceversa when too many times the algorithm has incremented the row number (or incremented the column number to many times in a row), the maximum number of times a decision may be repeated is another free parameter of the algorithm. In practice, the "on-line time warping" algorithm tends to deviate from the optimal path since the error is cumulative, this algorithm may completely loose track of the music.

In [2] Hidden Markov Models (HMMs) were used as the alignment tool. An HMM is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols [8]. Finding HMM's parameters given a sequence of observations is a problem known as training, this problem is solved by the Baum-Welch algorithm [9]. Using an HMM (e.g. for tracking purposes) is the so called "evaluation problem" which is solved by the forward or the backward procedure [10]. Finally the "Viterbi" algorithm is used to try to discover the hidden part of the model (e.g. which state should be connected to which states) unfortunately some critical decisions has to be made by the designer of an HMM, mainly the number of states.

Since alignment is an optimization process, one interesting proposal is to use particle filters and swarm optimization to produce the alignment, as described in [1].

The rest of the paper is organized as follows: In the next section we explain our novel technique for real time tracking of musical performances, there we provide details both for the feature extraction module (audio-fingerprint determination) and for the proximity index that we use instead of any alignment method (the traditional approach for tracking musical performances). In Section 4 we describe the test set, the experiments performed and results obtained, based on which we arrived to some conclusions written in Section 5 where we also propose some future extensions to this work.

3 Our Proposal

State of the art approaches use only local information for tracking performances on-line. This implies a cumulative error, a misalignment in time t_0 cannot be recovered in any $t > t_0$. The inductive hypothesis for dynamic time warping or hidden Markov models is that the alignment is correct from all the previous time frames and use this information to align the current frame. Even if such hypothesis allows for designing of fast algorithms, it is more natural to assume nothing about the past, thus allowing recovery from any possible previous error.

We propose in this paper to improve real time tracking of audio by the novel idea of using a proximity index instead of an alignment tool, using not only local but global information. We combine this idea with the use of an audio-fingerprint of our own design. This two improvements create a very powerful tool able to recover from past alignment errors, it is very fast and even tolerates noise on both the score and the online rendition side.

As we explained in Section 2, any musical performance tracking system has to extract features from the audio signal, we will now explain how we process the signal to determine the more convenient audio signature, the Multi-Band Spectral Entropy Signature which is by the way of our own design and was adapted for the issue of this paper.

3.1 Multi-Band Spectral Entropy Signature

In [11], the extraction of a very robust audio-fingerprint based on instantaneous entropy computed in frequency bands was used for retrieving music by content. The same audio signature was successfully used for automatic radio broadcast monitoring in [12] with excellent results. We adapted our Multi-Band Entropy Signature (MBSES) for the issue of tracking musical performances on-line, the adapted MBSES is determined as follows:

1. The signal is processed in frames of 185 ms, this frame size ensures an adequate time support for entropy computation. The frames are overlapped by 3/4 (75%), therefore, a feature vector will be determined every 46 ms approximately
2. To each frame the Hann window is applied and then its Discrete Fourier Transforms is determined.
3. Shannon's entropy is computed for the first 24 critical bands according to the Bark scale (frequencies between 20 Hz and 7700 Hz). To compute Shannon's entropy, equation 1 is used. σ_{xx} and σ_{yy} also known as σ_x^2 and σ_y^2 are the variances of the real and the imaginary part respectively and $\sigma_{xy} = \sigma_{yx}$ is the covariance between the real and the imaginary part of the spectrum.

$$H = ln(2\pi e) + \frac{1}{2} ln(\sigma_{xx}\sigma_{yy} - \sigma_{xy}^2) \tag{1}$$

4. For each band, decide if the entropy is increasing or not (e.g. compare with previous frame's band). Equation (2) states how the bit corresponding to band b and frame n of the signature is determined using $H_b(n)$ and $H_b(n-1)$ (The entropy values of frames n and $n-1$ for band b respectively). Only 3 bytes for each 32 ms of audio are needed to store this signature.

$$F(n,b) = \begin{cases} 1 \; if \;\; [H_b(n) - H_b(n-1)] > 0 \\ 0 \; Otherwise \end{cases} \tag{2}$$

After computing the MBSES in each overlapped audio frame the song is represented as a binary matrix with 24 rows (24 bands of Bark) and a number of columns that depends on the duration of the song.

The MBSES corresponding to approximately one second of audio (1110 ms to be precise) is a binary matrix of 24 rows and 24 columns. A single column corresponds to 46.25 ms. The MBSES of the score can be seen as a sequence of 24x24 binary matrixes. Our purpose is to search the current one-second of audio in the online rendition in every position of the score.

The training database stores the audio-fingerprint of each one-second segment and the location of the segment inside the score. We will search for the k-nearest neighbors (KNN) of the current one second segment of the rendition in the score song, and will select among those the closest in time. The above procedure is necessary because it can be the case the current segment correspond to a chorus or a repeating segment of the score song.

The next step consist in speeding up the k-nearest neighbor searching, which can be done using a metric index.

3.2 Metric Indexes

A metric index organize a data set equipped with a distance function (a metric space) to quickly answer proximity queries. We are interested in quickly answering KNN queries in the metric space defined by all the possible segments of the score song. For the proof of the concept detailed in this work we selected the Burkhard-Keller tree or BK-tree implemented in the SISAP library [13,14]. This data structure is well suited for our task.

Fig. 1 shows a BK-tree built from an hypothetical 14-second MBSES shown at the top of Fig. 1, Fourteen binary sub-matrixes are extracted and added to the BK-tree, each correspond to one second of audio and are referred to as a,b,..n. For the sake of space in Fig. 1 a,b,..n are 6x6 binary matrixes instead of the real 24x24 matrixes that result from extracting MBSES from of one-second excerpts.

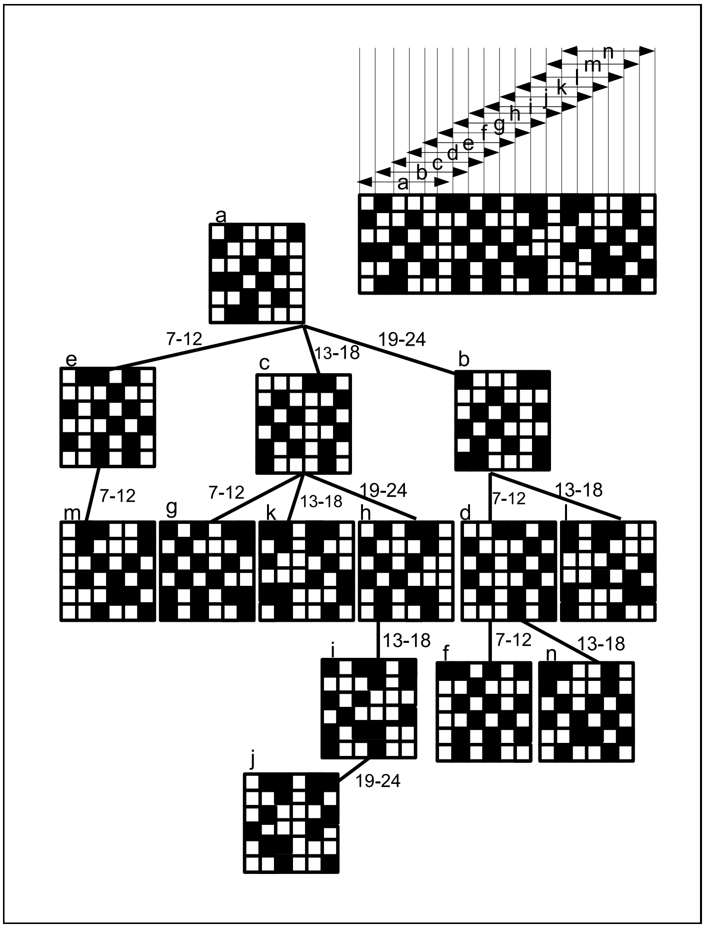


Fig. 1. One-second excerpts are indexed using a BK-tree

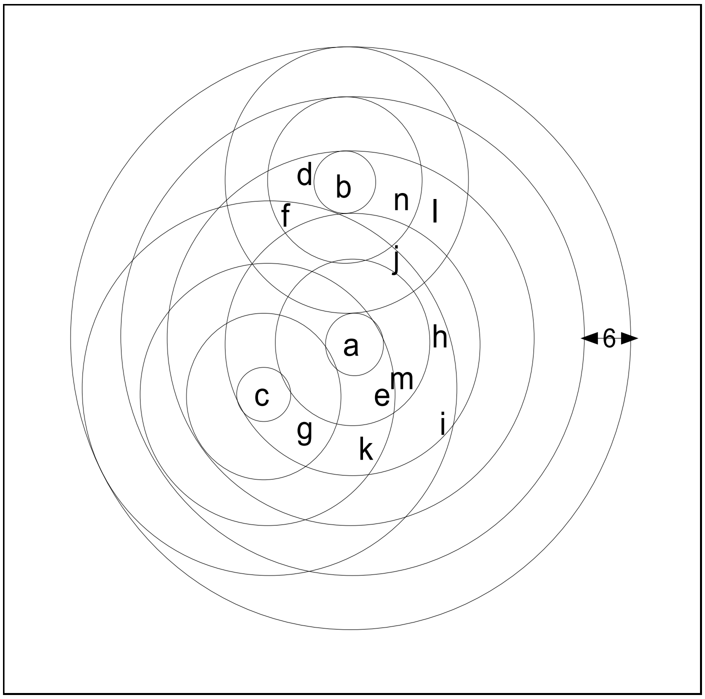


Fig. 2. Metric space related to the BK-tree shown in Fig. 1

The first submatrix read (Matrix a) becomes the root of the BK-tree, the second one is compared to the root using the Hamming distance which turns out to be 21 (21 different bits), so it becomes a son of the root under the link 19-24 since $19 \leq 21 \leq 24$, the third matrix read (matrix c) is next inserted, so it is compared to the root, its Hamming distance to the root is 17 so it becomes another son of the root under link 13-18 since $13 \leq 17 \leq 18$. To insert matrix d, it is first compared to the root, since its distance to the root falls in the rank 19-24 and there is already a node there (matrix b), then matrix d has to be compared with matrix b, it is then added as a son of matrix b under link labeled 7-12 since $7 \leq Hamming(b, d) \leq 12$ (In fact, Hamming(b,d)=12). The rest of the matrixes are inserted in the BK-tree the same way.

For the BK-tree in the example of Fig. 1 the bucket size in the nodes is $bs = 6$, that is, the maximum number of children of a node, also, the ring width is $s = 6$, that is the range of distances grouped in a single child of a node. To understand that, observe Fig. 2, the root of the BK-tree from Fig. 1 (sub-matrix a) is located at the center of the metric space. Six rings are shown around sub-matrix “a”, each one has a width of $rw = 6$. Any sub-matrix must lie inside one of these rings since the maximum Hamming distance between two $6x6$ binary matrixes is 36. Any sub-matrix that is a descendant of a son of the root lies inside the same ring, therefore, sub-matrixes “e” and “m” lie inside the same ring, also sub-matrixes “c”, “g”, “k”, “i”, “h”, and “j” conform a BK-subtree and lie inside the same ring. Any node that is a son of the root is itself the root of a BK-subtree, then they are at the center of another set of rings, observe for example sub-matrixes "n", "d" and "f", they lie inside the same ring centered at sub-matrix "b" and since these sub-matrixes along with sub-matrix "l" conform a BK-subtree they all lie inside the same ring centered at "a". To find the nearest neighbors in the Bk-tree, the rings whose distance to the center is less and less near the distance between the query and the center.

4 Experiments

For 62 songs or masterpieces we were able to obtain another musical rendition of it, for example for *Beethoven's Symphony Number 5 in C minor* performed by the *Berliner Philharmonische Orchester* conducted by Karajan we obtained the *Viena Philarmonic Orchestra* version conducted by Kleiber, such collection of songs and masterpieces conformed the data set for our tests.

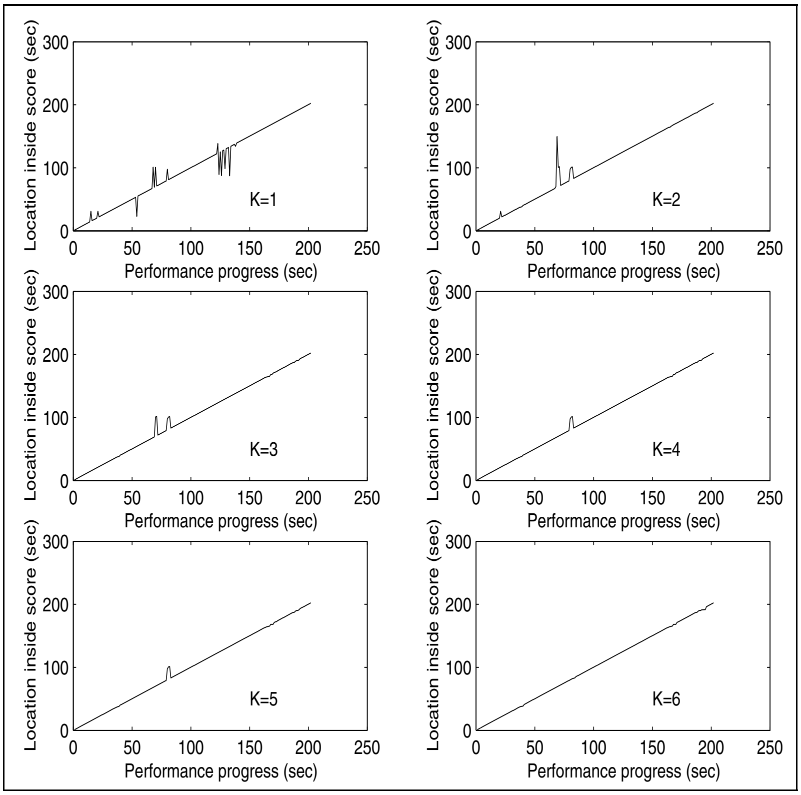
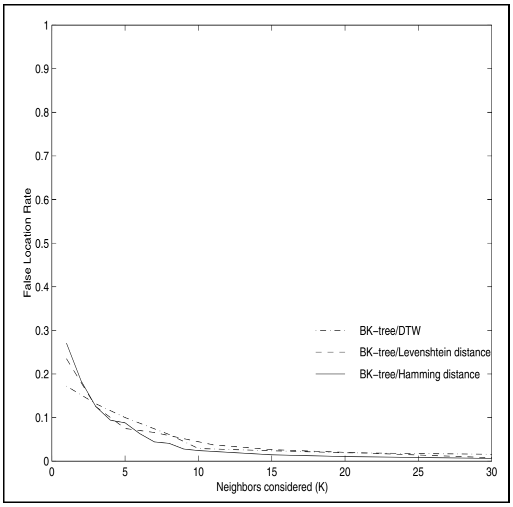


Fig. 3. Tracking of a performance of "All my loving" (The Beatles) for different values of K (Number of nearest neighbors considered). Some tracking failures can be observed for low vales of K.



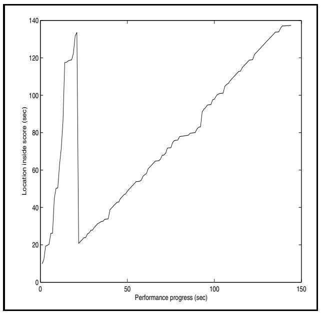


Fig. 4. The False Location Rate falls as K increases (left). An example of the recovery after a false start in *The Nutcracker* using only one nearest neighbor (right).

For every pair of musical performances of each song or masterpiece in our collection, we took the first one as the score and the second one as the rendition. We extracted and indexed the MBSES of the score using the BK-tree with a bucket size of $b = 24$ (the maximum number of children of a node) and a ring width of $s = 24$ (the range of distances grouped in a single node).

For the online tracking we proceed as follows: For every second of audio we extract its MBSES and then search for the K-nearest neighbors of it in the score using the closest one in time. We then read the next second of audio and repeat the process until the end of the performance. Fig. 3 shows the tracking of a performance of the Beatles' song ("All my loving"), the curves shown in Fig. 3 informs for each second of the performance where it was located in the score, normally, if both the score and the performance are very similar in duration as they are in the case of Fig. 3. We observe some peaks in Fig. 3 for low values of K, these peaks are in fact tracking failures since they imply a jump from a tracking position to another position that is not very near, we call this "false locations". We repeated the tracking for different values of K and observe that such false locations disappear as K increases, in the example shown in Fig. 3 the tracking failures disappear for $K = 6$. Observe that the system recovers almost immediately from false locations so they must not be considered has if the system lost track of the music. An extreme example of this recovery is shown in Fig. 4 (rigth).

Repeating the process described above for the rest of the musical performances of our collection we determined the False Location Rate (FLR) for values of K varying from 1 to 30. A false location is considered to appear when the short audio signal of a one-second length is located too far away in time from the previous tracked piece of audio. The FLR is computed using equation 3

$$FLR = \frac{False\ locations}{False\ locations + True\ locations} \tag{3}$$

In Fig. 4 (left) the FLR resulting from tracking every second musical performance of our collection and then varying the number of nearest neighbors considered (K). We indexed the metric space representing the score using three different distances. The first was the Hamming distance counting the number of bits different in each 24x24 matrix. The second was the DTW distance, and the third was the Levenshtein distance measuring the minimum number of insertions, deletions and substitutions to make the matrices equal; we used a substitution weight of 2, and the insertion and deletion weights were normalized between 0 and 1 using the Hamming distance between frames. We used up to 30 nearest neighbors for the global alignment. Using either of the three distances we were able to lower the false location rate arbitrarily. Since the Hamming distance is the cheaper one of the three, we believe it should be the one we should select to work.

4.1 Avoiding False Locations

For all practical purposes false locations should not lead to execute actions, that is they should be avoided. False locations appear due to the fact that none of the K nearest neighbors occur near the current tracking position in time. A simple way to prevent undesirable actions to take effect as a result of a false location occurrence can be stated like this:

When none of the K nearest neighbors occur near the current tracking position then the tracking position in the score should not move as if the last short segment of audio had not been received.

Once this modification was made to our tracking system the peaks such as those in Fig. 3 no longer occur, not even for K=1.

4.2 Time Analysis

Two steps are required to find the location inside the score of a one-second segment of audio taken from the tracked musical performance. The first step consists in determining the MBSES (the audio-fingerprint) of the audio signal, the second step is the search of the K nearest neighbors of the 24x24 matrix extracted on the previous step using the BK-tree for that purpose. Using a Dual-core 1.46 GHz pentium Laptop with 1GB of memory, 130 milliseconds are needed to accomplish step 1 and only 10 milliseconds for step 2. Building the BK-tree is performed prior to the tracking process and so its timing is not critical. A real time application such as those discussed in the Motivation section would use approximately 140 ms for every second of audio tracked without any parallelization.

5 Conclusions and Future Work

We successfully performed tracking of musical performances using a robust audio-fingerprint by searching in a metric space using BK-trees as proximity indexes. This approach does not require an iterative training process as Hidden Markov Models (HMMs) where the ideal number of states is unknown. Like HMMs we take advantage of the fact that we know the score a-priori (unlike on-Line DTW). It is very important to remark that our approach is a global alignment tool not accumulating error and able to recover from false locations. A single error cannot be considered has if the system completely lost track of the music when this false location appear. Finally our approach has the additional advantage that the tracking could be started at any time, no alternative approach is capable to begin tracking when the musical performance has already started for example when the tracking system is turned on too late.

There are other alternative metric indexes that should be considered when the metric space is larger than only one song. This can be very useful when a complete collection of audio is indexed instead of a single song. We will investigate the scalability issues of metric indexes for this particular application.

References

1. Sethares, W.A., Morris, R.D., Sethares, J.C.: Beat tracking of musical performances using low level audio features. IEEE Transactions on Speech and Audio Processing (2) (March 2005)
2. Cano, P., Loscos, A., Bonada, J.: Score-performance matching using hmms. In: ICMC 1999, Audiovisual Institute, Pompeu Fabra University, Spain (1999)
3. Orio, N., Déchelle, F.: Score following using spectral analysis and hidden Markov models. In: Proceedings of the ICMC, pp. 151–154 (2001)
4. Orio, N., Lemouton, S., Schwarz, D.: Score following: state of the art and new developments. In: Proceedings of the, conference on New Interfaces for Musical Expression, National University of Singapore, p. 41 (2003)
5. Sakoe, H., Chiba, S.: Dynamic programming algortihm optimization for spoken word recognition. IEEE Transactions on Acoustics and Speech Signal Processing (ASSP), 43–49 (1978)
6. Dixon, S.: Live tracking of musical performances using on-line time warping. In: 8th International Conference on Digital Audio Effects (DAFx 2005), Austrian Research Institute for Artificial Intelligence, Vienna (September 2005)
7. Dixon, S., Widmer, G.: Match: A music alignment tool chest. In: 6th International Conference on Music Information Retrieval (ISMIR), Austrian Research Institute for Artificial Intelligence, Vienna (2005)
8. Rabiner, L., Juang, B.: An introduction to hidden markov models. IEEE ASSP Magazine 3(1), 4–16 (2003)
9. Bilmes, J.A.: A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, Department of Electrical Engineering and Computer Science U.C. Berkeley (April 1998)
10. Rabiner, R.L.: A tutorial on hidden markov models and selected aplications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)

11. Camarena-Ibarrola, A., Chavez, E.: On musical performances identification, entropy and string matching. In: Gelbukh, A., Reyes-Garcia, C.A. (eds.) MICAI 2006. LNCS (LNAI), vol. MICAI, pp. 952–962. Springer, Heidelberg (2006)
12. Camarena-Ibarrola, A., Chavez, E., Tellez, E.S.: Robust radio broadcast monitoring using a multi-band spectral entropy signature. In: 14th Iberoamerican Congress on Pattern Recognition, pp. 587–594. Springer, Heidelberg (2009)
13. Figueroa, K., Chávez, E., Navarro, G.: The sisap metric indexing library, `http://www.sisap.org/library/metricspaces.tar.gz`
14. Burkhard, W.A., Keller, R.M.: Some approaches to best-match file searching. ACM Commun. 16(4), 230–236 (1973)

Recognizing Dactylogical Symbols with Image Segmentation and a New Differentiated Weighting Scheme

Laura Jeanine Razo Gil, Salvador Godoy-Calderón, and Ricardo Barrón Fernández

Centro de Investigación en Computación (Center for Computing Research), IPN, Av. Juan Dios Batiz, s/n, Zacatenco, 07738, Mexico City, Mexico
sophia_ljrg@hotmail.com, sgodoyc@gmail.com, rbarron@cic.ipn.mx

Abstract. This paper contains the theoretical mechanisms of the techniques for non-convex shape recognition, through the use of contour chains and differentiated weighting scheme on them. As an application example, we use a set of digital images that represent the various symbols contained in the dactylogical alphabet. First we introduce the reader to the many pre-processing and segmentation techniques applied to the set of images. Later on, we describe the use of direction codes to code the symbols' contour. Finally, a novel differentiated weighting scheme is incorporated to an ALVOT-type algorithm, and then used for the supervised classification (identification) of the various symbols within the image set. The proposed methodology is then evaluated and contrasted through a series of experiments.

Keywords: Dactylogical alphabet, contour chains, ALVOT algorithms, differentiated weighting diagram.

1 Introduction

The techniques of pattern recognition with contour chains suggests the use of sophisticated procedures (Fourier transform, calculation of general momentums, generation of grammars, Markov models, etc.) that imply operations, formulas and domain changes very complex [6].

This methodology is based on a Simple Voting Algorithm (ALVOT) [10] which is designed with a differentiated weighting scheme on contour chains, extracted from an image, that work in the same space of representation this mean, does not require any domain change.

The originality, simplicity and efficiency with which the chains are manipulated, results in an innovative and potential proposal to recognized non-convex shapes. To show the procedure and construction of this methodology we use, as an example, images that represent each of the letters of dactylogical alphabet.

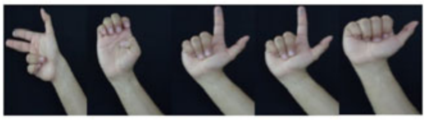


Fig. 1. The word "Hello"

To contrast our approach with respect to similar work in regard to the use of chains on the contour to recognize various forms are [13] and [14]. These proposals manipulate strings with methods that include changes of space and definition of more complex models. Also, There are different studies that have proposed solutions to the problem of recognizing the symbols of the dactylogical alphabet [1], [2], [3], [4] and [5]. Nevertheless, the importance of this proposal, compared to the formerly mentioned ones, is the definition of a classification model based on a differentiated weighting formula for the contour chains that represent, in this case, the different positions of the dactylogical alphabet.

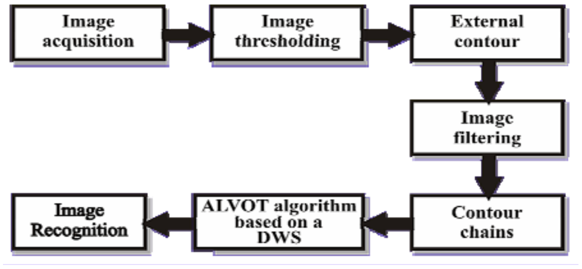


Fig. 2. Block diagram of the classifier system

This work contains a short introduction of the problem defined herein. Section 2 shows the pre-processing and segmentation stages. Section 3 describes the process for extracting features and classifying. Section 4 explains the set of experiments made to measure the efficiency of the recognition. Finally, section 5 shows the conclusions of this work.

2 Pre-processing and Segmentation

The processing and segmentation techniques applied to alphabet images were the following: identify and uniformize of image background; Otsu method to threshold the image; morphological operations to obtain the external contour; application of a neighboring filter 4 to slim down the contour and eliminate noise.

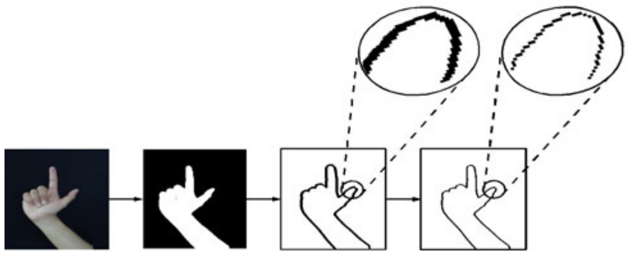


Fig. 3. Pre-processing and segmentation of letter "L"

3 Classification Methodology

The use of direction codes to express the contour is a process that is part of the classification methodology, because these codes generate a contour chain which is a representation pattern of each of the different hand positions. An aspect that must be highlighted is that contour chains are operated in their original representation space. This way, the comparison between codes is direct, thus avoiding the calculation of transformations to other spaces, which would be more complex.

Regarding the classification algorithm, the following is worth mentioning: it is supervised; it uses a simple voting algorithm, and what is even more meaningful; it employs an ad hoc differentiated pondering schematic representation of the contour chains.

The following paragraphs show a detailed explanation of each process carried out in this methodology.

3.1 Direction Codes and Contour Chains

A sequence of direction codes generates *contour chains*, which univocally determines the contour of an image. Direction codes represent the contour of an image by means of numeric values, as indicated by Figure 4. Each code represents, either the orientation of a set of points that form a straight line, or that of a set of segments with different length and direction [9].

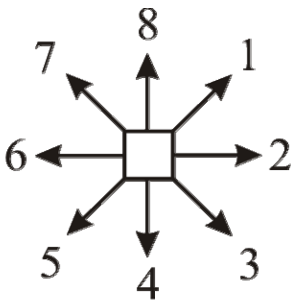


Fig. 4. Direction Codes (8-neighbouring system)

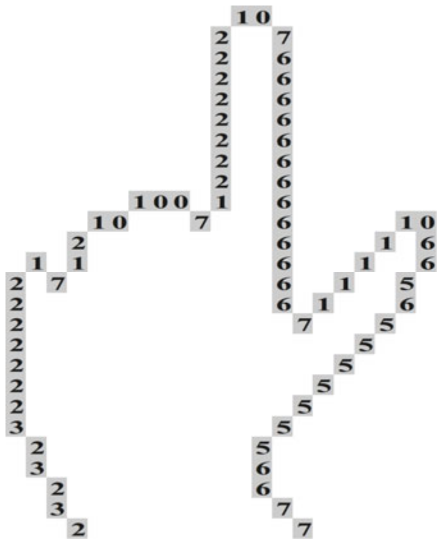


Fig. 5. Contour chain for letter "L"

Address codes are expressed without differences in direction between adjacent points due to the tolerance to rotation presented by the classification algorithm to some degree. (see section 3.3).

3.2 Regions and Segments

A fixed number of portions with the same length, called *regions,* are extracted from the contour chains. Each region is separated into three zones: initial, middle and ending. Likewise, the values coded for each zone are averaged to get three-directional values for each region.

When averaging the different zones of the regions we are able to: generate fixed-length contour chains, offset the high sensitivity to noise that is present in address code chains and obtain the space trend of the address values present in that portion of the analyzed contour. At the same time, taking a fixed number of regions based on the contour chain allows us to solve the problem of scale invariance.

Regions are gathered into sets called *segments*, which objective is to identify the more and the less significant areas in the contour chain which describes the different alphabet positions. Contrary to what happens in regions, a contour chain does not always have the same number of segments.

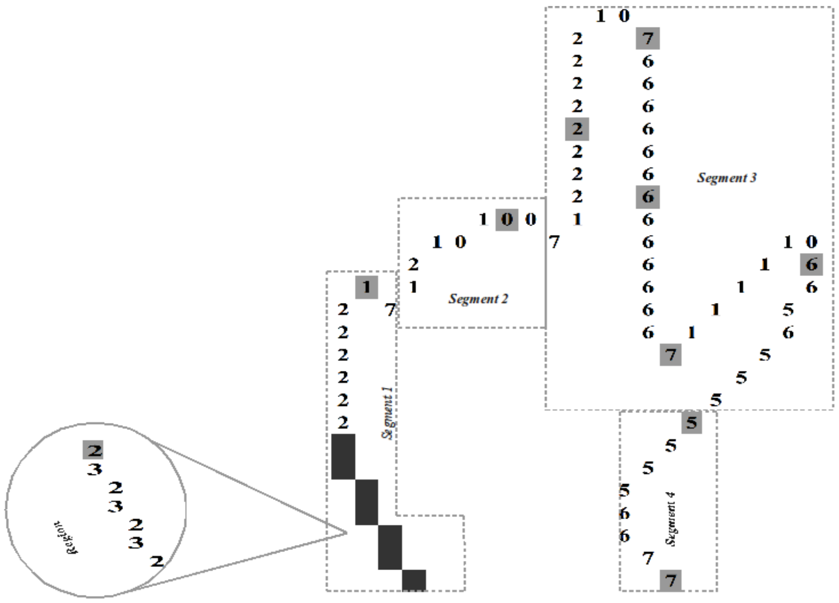


Fig. 6. Regions and segments of the contour chain of the "L" letter

The above figure is an example of the contour chain corresponding to letter "L" with 10 regions (the gray shaded address codes indicate the end of a region) and 4 segments (framed dotted).

3.3 Classification

The algorithm used to classify the images of the alphabet is a variant of a voting scheme (ALVOT) and shows three important characteristics:

- It is a simple voting scheme because the pattern under classification is compared to each and every one of the patterns contained in the supervision sample.
- The comparison between contour chains is made term by term.
- It includes a weighting scheme differentiated by class. This means that the difference between two segments that belong to different chains will be weighted with a different value, depending on the class to which the pattern under study will be compared.

The necessary elements for executing the classification algorithm are explained below.

3.3.1 Comparison Criterion for Direction Codes

The calculation of the similarity/difference between contour chains is based on the comparison of direction codes. This comparison is made through a function called Comparison Criterion (of a difference) between direction codes that give as result a numerical value that indicates the degree of difference between two direction codes that have been compared. This value is in the [0:1] range, where: 0 represents the minimum difference (identical direction codes), and 1 represents the maximum difference (opposite direction codes). Since the comparison criterion is symmetrical[1], and it has 8 different available directions, it is only possible to have three different comparison values. (see Figure 7)

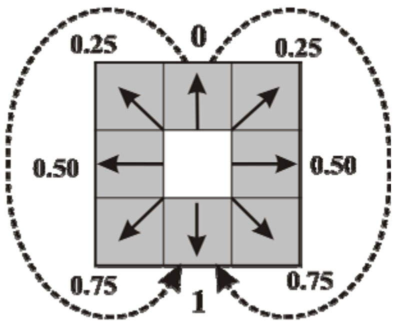


Fig. 7. Dynamics of the Comparison Criterion

3.3.2 Dissimilarity Function between Contour Chains

To find the difference between two patterns described by contour chains, the following expression is used:

$$f(A,B) = \frac{1}{S}\sum_{j=1}^{S}\left(W_j\left[\frac{1}{V_j}\sum_{i=1}^{V_j} Cc(A_i, B_i)\right]\right) \tag{1}$$

Where:

A: is the pattern to be classified,
B: is a pattern in the supervision sample,
S: is the number of segments of the contour,
V_j: is the amount of direction codes in segment j,
W_j: is the weight assigned to segment j in the class, to which B belongs,
Cc: is the comparison criterion between direction codes.

3.3.3 Choice of Weighting Scheme

A visual analysis of the universe of studied images produced the following remarks:

- All symbols have a unique contour.
- The contour of the wrist area gives information of very little relevance because it is present in all of the alphabet images.
- The contour area at the hand palm level contains more discriminating information because its position and orientation change for some hand positions.

[1] Remember that a comparison criterion is ***symmetrical*** if it always assigns the same value for two direction codes with the same separation (as shown in Fig. 8), either clockwise or counter-clockwise.

– The contour area of the fingers is highly relevant because each one of the finger configurations is different for each symbol in the dactylological alphabet.

In order to get a *weighted differentiation scheme,* an *informational weight (real value in [0,1])* is assigned to each segment identified in each class of the supervision sample depending on its relevance according to the execution of each symbol. This informational weight is different for each class in the supervision sample.

Figure 8 illustrates the weighting of segments from letters W and Q. The contour parts framed with a solid line are elements of little significance; framed dotted we have the elements which are more significant for discrimination and in framed dashed we have the most significant elements.

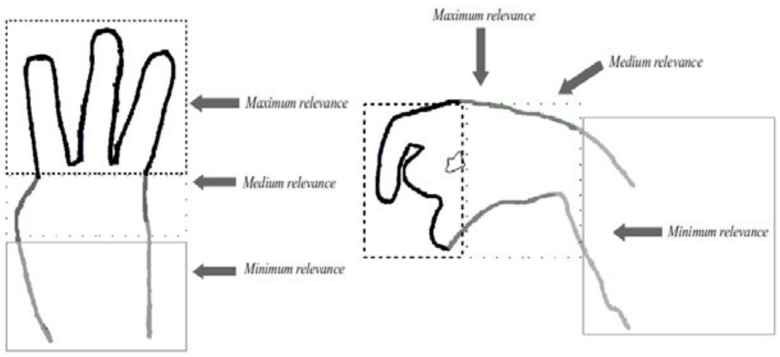


Fig. 8. Relevance of the segments of letters W and Q

To calculate weights are commonly used, for simplicity, ALVOT. There is another alternative that is Testors Theory, however testors calculation is a problem of exponential complexity and constantly looking for new options [12].

3.3.4 Learning Stage

The learning stage consists in obtaining the difference value between the pattern under classification with all of the patterns present in the supervision sample, by using the previously defined difference function (see equation 1).

3.3.5 Voting Stage

The voting function shows the amount of evidence available to indicate the class to which the pattern to be classified belongs. In this algorithm, the vote for pattern P to be in class Cj ($V_{C_j}(P)$), is the average of the differences between pattern P and all the patterns contained in class Cj of the supervision sample. The following expression defines the Vote Function:

$$V_{C_j}(P) = \frac{1}{|C_j|} \sum_{X \in C_j} f(P, X) \tag{2}$$

where:

P: is the pattern to be classified.
$|C_j|$: is the cardinality of the C_j class.
$f(P,X)$: is the calculated difference between the P and $X \in C_j$. patterns.

3.3.6 Rule for Obtaining the Solution

This procedure consists on defining, based on the information obtained from the votes, the class(es) to which the pattern to be classified will be assigned, as well as its degree of membership in these classes. Classes are modeled as crisp sets so the membership of a pattern to any class can also take the values: 1 (if it belongs) and 0 (if it does not belong). The pattern is assigned to the class in which it has the minimum vote. In case of a tie in the voting of two or more classes, the rule for solution includes a specific procedure to solve these cases:

1. Calculate a holotype[2] for each of the classes in the tie.
2. Compare the pattern to be classified with each of the calculated holotypes.
3. Assign the pattern to the class which holotype shows the smallest difference to the original pattern.

In case that a tie persists, calculated the centroids for each of the classes in the tie, assign the pattern to the class which centroid shows the smallest distance to the original pattern.

4 Experimental Results

As a basis for the recognition methodology proposed in the previous section, a set of numerical experiments were performed. These experiments include tests regarding tolerance to rotation, and size of the supervision sample. At the end of each experiment a discussion of the expected results is included.

To form our initial sample of application, a total of 375 images from 15 different people were taken. Only hand positions which did not represent movement were captured. We used the Mexican dactylogical alphabet [11]. At the classification stage, the number of regions forming the contour chain was 40.

The bank of images was divided into two disjoint sets, the first one plays the role of supervision sample and the second one is the control sample.

4.1 Experiment #1: Initial Recognition without Differentiated Weighting

This experiment was set to find the recognition effectiveness when applying the classification methodology without considering the differentiated weighting on the contour chains. Therefore, we took a supervision sample with 192 images and a control sample with 100 images. Table 1 shows the results obtained from this experiment.

[2] The holotype of a class is defined as the pattern whose average similarity with the rest of the elements in the same class is optimum.

Table 1. Results yield by experiment #1

Supervision Sample	Control Sample	% Right answers	% Wrong answers
192	100	48	52

4.1.1 Discussion

Results yielded by this first experiment were not satisfactory to conclusively recognize the alphabet images. The relevance of weighting the contour chain is shown in the classification efficiency; the algorithm faces some pattern confusion due to the fact that all the areas corresponding to the contour chain have the same relevance.

4.2 Experiment #2: Initial Recognition with Differentiated Weighting

This experiment will give us an approximate idea of the accuracy achievable in an actual recognition context. In case unsatisfactory results are obtained, the necessary adjustments must be made.

The supervision sample has 192 images (8 images of each symbol made by different people) and the control sample has a total of 100 (4 images of each letter).

Table 2. Results yield by experiment #2

Supervision Sample	**Control Sample**	**% Right answers**	**% Wrong answers**
192	100	80	20

It is important to mention the fact that even if the great majority of the works that have been written about this topic the ratio of patterns contained in these two samples is 80-20 (80% patterns in the supervision sample and 20% in the control sample), in this case we made an a priori assumption: that given the characteristics of the classification algorithm having twice as many supervision samples compared to the control elements was enough (which yields an approximate 65%-35% ratio). In Table 2 we show the recognition ratio achieved by Experiment #2.

4.2.1 Discussion

In spite of the fact that the supervision and control samples do not have the typical 80%-20% ratio, results obtained were satisfactory. Even when the methodology does not get the right answer in some cases, if the number of cases is increase in both samples, a better recognition will be achieved.

There is an interesting peculiarity that is worth mentioning: some letters are treated in a very similar way, like “M” and “N” or “G” and “H” [11]. The consequence is that most of the patterns classified wrongly, fall into very similar classes. This can be solved by adjusting the differentiated weighting schematic between them to achieve more efficiency in the discrimination.

4.3 Experiment #3: Tolerance to Rotation

This test is aimed at showing tolerance to image rotation through two representations of the chain code. The first representation consists of taking the difference between every two points of the chain to make it invariant to rotation and the second one takes the address codes according to the proposed methodology for solution.

Three image banks were used: two control samples and a supervision sample. The first control sample contains artificially rotated images and the second one has images of new hands captured with different rotation changes. Both samples were tested with differentiated and with not differentiated address codes.

4.3.1 Artificially Rotated Images

In this test, two groups of images were artificially rotated: the ones misinterpreted when rotated ("C", "G", "L", "P") and the ones which are not misinterpreted when rotated ("A", "D", "O", "W"). Images were rotated starting from -45^o to $+45^o$, with 2^o increases. The recognition was made with differentiated and not differentiated code chains. Table 3 shows the tolerance level achieved, with undifferentiated code representation.

Table 3. Table with the results obtained in experiment #3 (Artificially rotated images with undifferentiated address codes)

Supervision Sample	Control Sample	Test Groups	Rotation	Tolerance
192	138	Letters not misinterpreted when rotated (A, D, O, W).	±45° 2° increases	±21°
192	138	Letters misinterpreted when rotated (C, G, L, P).	±45° 2° increases	±21°

Next table shows the achieved tolerance level with differentiated code representation.

Table 4. Table with the results obtained in experiment #3 (Artificially rotated images with differentiated address codes)

Supervision Sample	Control Sample	Test Groups	Rotation	Tolerance
92	138	Letters not misinterpreted when rotated (A, D, O, W).	±45° 2° increases	None
192	138	Letters misinterpreted when rotated (C, G, L, P).	±45° 2° increases	None

4.3.2 Images Captured with Rotation Changes

In this experiment we captured images of two letter groups. The first group is made of symbols that are confused with other symbols of the dactylogical alphabet when they are rotated ("C", "G", "L", "P") and the second one contains letter symbols that are not confused when they are rotated ("A", "D", "O", "W"). Rotation changes were made from 0^o to 90^o approximately [11].

Table 5 shows the results obtained from the application of recognition to control samples of groups that are misinterpreted when the hand rotates, and to those who don't. The pattern representation was made with undifferentiated direction codes.

Table 5. Table with the results obtained in experiment #3

Supervision Sample	Control Sample	Test Groups	Rotation	Tolerance
192	30	Letters not misinterpreted when rotated (A, D, O, W).	16	22
192	30	Letters misinterpreted when rotated (C, G, L, P).	11	21

Table 6 shows the results obtained when making experiments with both test samples. In this case, differentiated chain codes were used.

Table 6. Table with the results obtained in experiment #3 (Images captured with rotation changes and differentiated direction codes)

Supervision Sample	Control Sample	Test Groups	Rotation	Tolerance
192	45	Letters not misinterpreted when rotated (A, D, O,W).	5	21
192	30	Letters misinterpreted when rotated (C, G, L,P).	0	21

4.3.3 Discussion

In the literature, the difference between two points of the chain is frequently used in order to make the chain invariable with regard to rotation. Nevertheless, the methodology proposed herein uses non-differentiated direction codes. Experimenting with both representations was basic to determine that differentiated direction codes hamper the efficiency of the recognition. The ability of our methodology to support ±21° of tolerance to rotation in dactylogical symbol images, compared to the classical model, which shows very poor results, is worth noticing.

The failure of the classical model is due to the fact that the difference between every two points of the chain shows a smaller variation in the direction code values, making them become more similar and less differentiated. The opposite effect is obtained with the proposed methodology, in which there is no difference between direction codes, therefore allowing more variations in the regions that divide the contour.

The wrongly classified patterns that correspond to the hands captured with different rotations belong to letters expressed with bad spelling. The tests showed that rotating letters such as "C", "G", "L" and "P" more than 45° causes confusion, as in the case of a letter "C" rotated more than 45° which may be mistaken by a letter "Q".

The results achieved here exceed the expectations contained in the classification methodology. The range of tolerance grants enough flexibility to execute a dactylogical symbol with different rotations without causing any confusion or bad spelling.

4.4 Experiment #4: Sample Distribution

This experiment basically consists of calculating the minimum and maximum recognition percentages with different proportions in the supervision and control sample. This means that 6 supervision and control samples were created: in the first case, the supervision and control sample was formed with a 20%-80% proportion; the second was formed with a 40%-60% and in the third with a 20%-40%; The fourth one with a 40%-20%; the fifth one with a 60%-40% and the sixth one with 80%-20%. (see Figure 9).

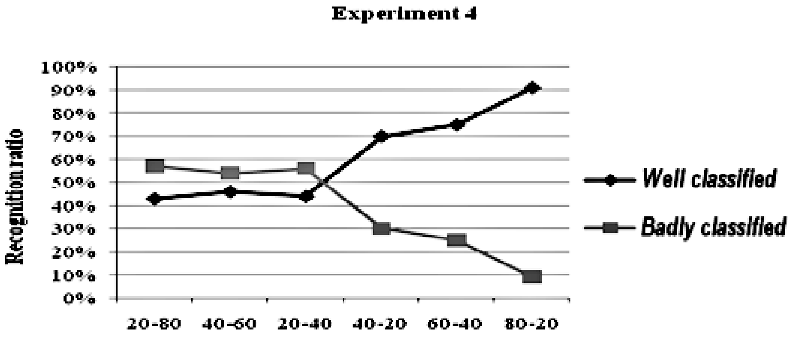


Fig. 9. Results of experiment #4

4.4.1 Discussion

The conclusion is that, the more learning is acquired from the supervision sample, the better the recognition efficiency of the images will be. The fact that the 80%-20% recognition showed the highest percentage of recognition compared to the 20%-80% one, which obtained the lower percentage, is worthwhile highlighting.

Two important aspects are worth noticing: the first one is that the higher the amount of elements in the sample, the better the learning will be. The other aspect underlines the quality of the experimental results, which were quite satisfactory, thus reasserting the effectiveness, originality, simplicity and robustness with which the methodology for solution faces the problem of recognition of the dactylogical symbols.

Lastly, it was not necessary to make a special experiment for the different hand sizes because the problem of invariance at a scale was solved by separating the contour chain into a fixed number of regions.

5 Conclusions

The first aspect to be highlighted is that an innovative criterion for comparing direction codes with an 8-directional schematic operating on its own representation scale was defined. The second remark is related to the great similarity of patterns that belong to different classes. Therefore, setting a differentiated weighting scheme on the more and the less meaningful parts of the contour of a dactylogical symbol is quite necessary. When we established this scheme as a part of ALVOT, we were able to increase the precision of recognition noticeably, at the same time keeping the known simplicity of an algorithm of the ALVOT family. Later on, the importance of using direction codes without there being a difference between them, like in the classical model, was recognized. This was due to the fact that a comparison criterion and an ad hoc similarity function that treats direction codes in a simple way, was developed.

Another aspect to be considered is the tolerance to scale solved through the division of the contour chains into a fixed number of regions. Also, tolerance-to-rotation levels are noticeable due to the fact that we do not need to rotate the hand too much when making a dactylogical symbol, because this would mean that the letter represents a spelling mistake.

The next thing to be acknowledged is that if we add atypical elements, such as the images of dactylogical symbols excessively rotated, a better learning is achieved, thus increasing the recognition efficiently remarkably.

Another thought is that the recognition efficiency completely depends on the weighting scheme, which does not represent weakness, but an advantage, because this methodology was created to use weightings that may be adjusted, improving recognition considerably.

Finally, and perhaps the most important item to be highlighted is that the recognition methodology may be generalized as a methodology for the recognition of non-convex figures, taking advantage of the differentiated weighting scheme and the representation through contour chains.

Acknowledgements. Work done under partial support of Mexican Government (CONACYT, SNI) and National Polytechnic Institute, Mexico (SIP Projects 20071432, 20082204, 20091249, 20100068; COFAA, PIFI).

References

[1] Jaime, S.M.: Sistema eficiente de reconocimiento de gestos de la mano (2000) (in Spanish)

[2] Jaime, L.I., del Rocío, R.B.M., Verónica, P.T.: Josefa. Sensor foto-eléctrico aplicado al movimiento de los dedos de las manos (2005) (in Spanish)

[3] Allen, J., Asselin, P., Foulds, R.: American Sign Language Finger Spelling Recognition System. In: IEEE 29th Annual Proc. of Bioengineering Conf. 2003, pp. 285–286 (2003)

[4] Seungki, M., Sanghyeok, O., Gyoryeong, K., Taehyun, Y., Chungyu, L., Yunli, L., Keechul, J.: Simple Glove-Based Korean Finger Spelling Recognition System. In: Gervasi, O., Gavrilova, M.L. (eds.) ICCSA 2007, Part I. LNCS, vol. 4705, pp. 1063–1073. Springer, Heidelberg (2007)

[5] Tabata, Y., Kuroda, T.: Finger Spelling Recognition using Distinctive Features of Hand Shape. In: Proc. of 7th ICDVRAT with ArtAbilitation, Maia, Portugal (2008)

[6] González, R.C., Woods, R.E.: Digital Image Processing, 3rd edn. Prentice-Hall, Englewood Cliffs (2008)

[7] Humberto, S.A.J.: Rasgos descriptores para el reconocimiento de objetos, Primera edición. IPN (2006) (in Spanish)

[8] MartinSanz, G.P., Gracía de la Cruz Jesús, M.: Visión por Computador, Imágenes digitales y aplicaciones, Segunda edición. Alfaomega (2008) (in Spanish)
[9] Darío, G.-A.M.: Reconocimiento de formas y visión artificial, 2nd edn. Addison-Wesley, Reading (1994) (in Spanish)
[10] José, R.-S., Adolfo, G.-A., Francisco, M.-T.: Enfoque Lógico Combinatorio al Reconocimiento de Patrones. I. Selección de Variables y Clasificación Supervisada, Primera edición. Ed. IPN (1999) (in Spanish)
[11] de Fleischmann María Esther, S.: Lenguaje Manual, aprendizaje del español signado para personas sordas, Segunda edición. Trillas, pp. 23–29 (in Spanish)
[12] de testores típicos, C., Jiménez, V., Ruiz-Shulcloper, J.: Memorias del I Taller Iberoamericano de Reconocimiento de Patrones. In: Arenas, A.G., Shulcloper, J.R., Azuela, H.S. (eds.) TIARP 1995, La Habana, Cuba, January 23-27, pp. 171–180. Instituto Tecnológico de Toluca, Metepec, México (1995) (in Spanish)
[13] Persoon and K. Fu. Shape Discrimination Using Fourier Descriptors. IEEE Transactions on Systems, Man, and Cybernetics 7 (1977)
[14] Bicego, M., Murino, V.: 2D shape recognition by Hidden Markov Models. IEEE Transactions on Pattern Analysis and Machine Intelligence (2002)

Selecting Candidate Labels for Hierarchical Document Clusters Using Association Rules

Fabiano Fernandes dos Santos[1], Veronica Oliveira de Carvalho[2], and Solange Oliveira Rezende[1]

[1] Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo (USP)
{fabianof,solange}@icmc.usp.br
[2] Instituto de Geociências e Ciências Exatas
UNESP - Univ Estadual Paulista
veronica@rc.unesp.br

Abstract. One way to organize knowledge and make its search and retrieval easier is to create a structural representation divided by hierarchically related topics. Once this structure is built, it is necessary to find labels for each of the obtained clusters. In many cases the labels have to be built using only the terms in the documents of the collection. This paper presents the SeCLAR (Selecting Candidate Labels using Association Rules) method, which explores the use of association rules for the selection of good candidates for labels of hierarchical document clusters. The candidates are processed by a classical method to generate the labels. The idea of the proposed method is to process each parent-child relationship of the nodes as an antecedent-consequent relationship of association rules. The experimental results show that the proposed method can improve the precision and recall of labels obtained by classical methods.

Keywords: label hierarchical clustering, association rules, text mining.

1 Introduction

The automatic organization of available textual information is a task that has received much attention in recent years. One way to organize knowledge and make its search and retrieval easier is to create a structural representation divided by hierarchically related topics. This structure can be obtained automatically by applying hierarchical clustering algorithms or can be constructed and maintained by domain experts, as in cases of online directories of Yahoo or Open Directory Project [12]. The more general knowledge of the collection is at the highest levels of the hierarchy and the more specific at the lower levels.

Once this structure is built, it is necessary to find labels for each of the obtained clusters. Labeling clusters is a common task in text mining and information retrieval. Generally, the methods find a list of discriminative terms, that are used to facilitate information retrieval or interpretation of each cluster [7].

In many cases the labels have to be built using only the terms in the documents of the collection. The simplest method is to select the most frequent words in the documents of the cluster. This list reveals the topic at a higher level, but can fail to depict specific details of the cluster [8].

The label selection methods found in the literature can be divided into methods based on frequency, as in the previously mentioned method, centroid-based methods, as proposed in [4], and methods based on probabilistic models ([3], [7], [8], [12]). These works consider each document as a *bag-of-words* and do not explore explicitly the relationship between the terms of the documents.

Thus, this paper presents the SeCLAR (Selecting Candidate Labels using Association Rules) method, which explores the use of association rules for the selection of good candidates for labels of hierarchical clusters of documents. This method generates association rules based on transactions built from each document in the collection, and uses the information relationship between the nodes of hierarchical clustering to select candidates for labels, as described in Section 3. Once selected, the candidates are processed by a classical method to generate the labels, obtaining the final set of labels for nodes in the hierarchy.

The paper is structured as follow: Section 2 presents a brief review of concepts used in this work and some methods found in literature. The proposed method is described in Section 3. Section 4 contains the results of the performed evaluation. Finally, conclusions and future work are presented in Section 5.

2 Related Works

In many problems, such as organizing a digital library, the hierarchical clustering of documents is already built. In these cases, a set of labels must be obtained for nodes in the hierarchy respecting the existing organization and using the information contained in the collection. The task of identifying discriminative words in each cluster can be viewed as an attribute selection problem [13].

An idea that is independent on the way the hierarchy is obtained and with the aim to avoid unnecessary word repetitions is presented in [8]. It starts from the root to the leaves of the tree, checking all the nodes. Each node receives a list of possible labels consisting of all terms of the documents associated with this node. In order to select the final list of labels, each term with frequency greater than zero is evaluated according to their degree of dependence or association with each child node. If the hypothesis of independence is accepted, then the word belongs to the parent node and is removed from the child. Otherwise, it is associated with the child nodes and removed from the parent node. One of the problems of this method is to guarantee the convergence of the difference distribution between the observed and expected values for χ^2 distribution, used in the tests when the expected values are very small (close to zero).

The authors of [7] proposed RLUM - Robust Labeling Up Method - based on the ideas of the method presented in [8]. This method solves more directly the issue of repetition of the labels and includes the use of a more robust statistical test that does not depend on empirical threshold estimation. Moreover, the goal is to distribute the terms along the hierarchy, avoiding unnecessary repetitions in the same branches, keeping the most generic terms in the high levels and the most specific terms in the lower levels. The RLUM is more robust and has a lower complexity than the method proposed in [8]. However, this method considers the documents as a *bag-of-words* and does not consider the relationships between the terms in documents.

The proposal described in [5] uses association rules as labels of non-hierarchical clusterings. According to the authors, the association rules carry an intrinsic semantic level, which is not possible to determine through a simple frequency count. The generated association rule is used as a label of the cluster, and the generation of rules also considers each document as a *bag-of-words*. Moreover, the association rules are easily understood and interpreted by a specialist as well as a normal user [6].

3 Proposed Method

Observing the characteristics of association rules and previous works, this paper proposes the SeCLAR - Selecting Candidate Labels using Association Rules. The general idea of the proposed method is to consider each relationship *parent-child* between nodes as the type of *antecedent-consequent* relations used in association rules. It is assumed that the parent node influences each of its child nodes, and thus selecting labels for the parent node should reflect this information.

The SeCLAR inputs are a hierarchical clustering H built from a set of documents D, a set of nodes N associated with H and a set of terms T previously selected for the set of terms in the collection. Set D is formed by documents used to build the hierarchy. In case of any preprocessing step of documents, like cleaning, standardizations and removal of words, the files obtained at the end of the process are used. Each node of set N has a set D' of documents associated with this node so that D' is a subset of D. Set T is formed by all terms in the documents of set D, as well as by the information of relationship between terms and documents, and by a measure of frequency associated with this relationship, usually represented by a attribute-value matrix. This measure can be, for example, the information about the absolute frequency of the term in the document.

The method presented in Algorithm 1 also considers the following parameters as input: (i) *window_size* is used to adjust the window size used to transform the documents into a set of transactions (details in Section 3.1); (ii) *measure* refers to objective measure that will be used to select a subset of association rules, implemented by Algorithm 3 (details in Section 3.2); (iii) K represents the maximum number of association rules selected in Algorithm 3 according to the measure chosen in Parameter *measure*; (iv) $minsup$ and $minconf$ are the values of minimum support and minimum confidence, respectively, used by the algorithm to generate association rules.

The process begins with the generation of sets of association rules associated with each node of the hierarchy (lines 1-4). The auxiliary algorithm *generate-association-rules* transforms these documents on transactions and obtains a set of association rules for this node (Section 3.1). Then the process of selecting candidates for labels is run (lines 5-8). The strategy to traverse the nodes (line 5) consists of exploring the most left child node not yet visited. If all child nodes have been explored, the execution continues processing the node visited. This strategy is important because it influences the result of the auxiliary algorithm *select-candidate-labels* described in detail in Section 3.2. It is possible to adopt other strategies to traverse the hierarchy, which were not evaluated in this work.

Algorithm 1. SeCLAR

```
Require: Hierarchical clustering H; set N={$n_1, ..., n_m$} of nodes associated with H; set
    D={$d_1, ..., d_k$} of documents associated with H; set T={$t_1, ..., t_j$} of terms; $window\_size$: window size; $measure$: an objective measure to select association rules; $K$: maximum number of
    association rules selected for each node; $minsup$ and $minconf$: the values of minimum support
    and minimum confidence to generate association rules.
Ensure: Hierarchical clustering $H'$ with candidate labels for each node.
1: for each node $n_i \in H$ do
2:     Retrieves the set $D' \subseteq$D of documents associated with the node $n_i$
3:     $R_i \leftarrow$generate-association-rules($n_i, D', window\_size, minsup, minconf$)
4: end for
5: for Visits the tree nodes from the leaves to the root, and for each node $n_i \in H$ do
6:     Retrieves the set $N' \subseteq$N of the children nodes of $n_i$
7:     $Labels_i \leftarrow$select-candidate-labels($n_i, N', measure, K$)
8: end for
9: $H' \leftarrow$remove-nodes(H)
```

Finally, the algorithm *remove-nodes* described in line 9 checks whether all nodes of the hierarchy have candidates for labels. If the set of candidates of any node is empty, we chose to remove this node from the hierarchy and their descendants are joined at the direct antecedent of the removed node according to process described in Section 3.3. The result obtained at the end of this process is used as input by a classical label selection method, as presented in Section 3.4.

The algorithms mentioned in lines 3, 7 and 9 will be described in detail in the following sections.

3.1 Generate the Association Rules for Each Node

It is necessary to transform documents associated with the node into a set of transactions to generate association rules for each node in the clustering. This paper assumes that for each node $n_i \in N$ it is possible to transform documents associated with this node in a set of transactions and, from this set, generate association rules. The method described in Algorithm 2 executes this transformation and obtains the set of association rules of the node n_i.

The input for this method are all preprocessed documents available in the node. For each document d_i, the words in the sequence they appear in the document are obtained (line 3). The i-th *word* found is added to the set which will form the new transaction. After that, the consecutive not repeated words is added into the set that will form the transaction. The process is repeated until the set has *window_size* size or until it is not possible to get words in the processed document (lines 4 the 11). The transaction formed is added to the set of transactions related to the node n_i (line 12), and the process runs until all documents are processed in the transactions. At the end of this process, the set of transactions owned by the node is processed by an algorithm to generate association rules (such as Apriori [1]), and the obtained set is stored (line 15). This conversion process using windows enables the exploration of the relationships among words in documents with different levels of granularity.

Algorithm 2. generate-association-rules Method

```
Require: $D' \subseteq$D of documents associated with the node $n_i$; $window\_size$: window size; $minsup$:
    minimum support value; $minconf$: minimum confidence value.
Ensure: The set $R_i$ of association rules of the node $n_i$
 1: $Transactions_i \leftarrow \varnothing$
 2: for Each document $d_k \in D'$ do
 3:     for i←1 to $|words\ in\ d_k|$ do
 4:         $transaction \leftarrow \varnothing$
 5:         $w \leftarrow i$
 6:         while $|transaction| <= window\_size$ AND $i + w <= |words\ in\ d_k|$ do
 7:             if $\{word_{i+w}\} \notin transaction$ then
 8:                 $transaction \leftarrow transaction \cup word_{i+w}$
 9:             end if
10:             $w \leftarrow w + 1$
11:         end while
12:         $Transactions_i \leftarrow Transactions_i \cup transaction$
13:     end for
14: end for
15: $R_i \leftarrow$ generate-rules($Transactions_i, minsup, minconf$)
```

3.2 Select Candidate Labels

After obtaining the set of association rules for all nodes, the candidate labels for each node n_i are selected. The proposal is to use association rules of each child node n_i as described in Algorithm 3.

Algorithm 3. select-candidate-labels Method

```
Require: $measure$: an objective measure to select association rules; $N'$: the children nodes of $n_i$
    $K$: maximum number of association rules to be selected for each node.
Ensure: The set of candidate labels associated with the node $n_i$
 1: $Labels_i \leftarrow \varnothing$
 2: if ($n_i$ is leaf) then
 3:     $Labels_i \leftarrow$ The current set of terms of $n_i$
 4: else
 5:     for $j \leftarrow 1$ to $|N'|$ do
 6:         Retrieves the set $R_j$ of association rules of the node $n_j \in N'$
 7:         TopRules←select-best-rules($R_j, measure, K$)
 8:         for $w \leftarrow 1$ to $|TopRules|$ do
 9:             Rule← $TopRules[w]$
10:             Antecedent← Rule.Antecedent
11:             for Each term ∈ Antecedent do
12:                 if {term}$\notin Labels_i$ AND {term} is not a candidate label of any descendent of
    $n_i$ then
13:                     $Labels_i \leftarrow Labels_i \cup$ {term}
14:                 end if
15:             end for
16:         end for
17:     end for
18: end if
```

Initially, the algorithm checks the node type to decide which strategy will be adopted. It is not necessary to select leaf nodes since they don't posses any child node. Thus, if the node n_i is a leaf node evaluated in the hierarchy, all terms are added to the list of possible candidates. If the node is not a leaf, the method *select-best-rules* in line 7 retrieves the K top association rules for each of the sets of rules from the child nodes of n_i. From this subset the items of the antecedent of each rule are selected and added to a set of possible labels of the node n_i (lines 8-17). The K top association rules are the ones which have the highest objective measure values.

The proposed method uses the items in the antecedent of the association rules of its child nodes to explore the relationship *antecedent-consequent* as mentioned in the beginning of Section 3. Thus, it is expected that the selected labels for a node are those that best describe the information contained in their children, exploring the existing hierarchical organization. The items in the consequent of the association rules are not used in the selection process, since Algorithm 3 is applied recursively to all nodes. It is important to note that a good choice for the selected labels depends on the objective measure.

The method does not allow a node to have as a candidate label a term that is already used by a descending not leaf node and, as a consequence, the terms are not replicated in the hierarchy (line 12). Therefore, it is possible that one or more nodes have at the end an empty set of labels. However, this restriction does not prevent the same label to be used in nodes of a same level or in different branches of the hierarchy.

3.3 Removing the Hierarchy Nodes That Contain No Labels

After applying the method presented in Algorithm 3 on all nodes of the hierarchy, it is possible that the set of labels of some nodes remained empty. In this case, the node is removed from the hierarchy, and their descendants are associated with the parent node of the removed node.

Algorithm 4. remove-nodes Method

```
Ensure: Hierarchy H updated
1: for each node $n_i \in H$ AND $n_i$ isn't root do
2:     if $Labels_i = \varnothing$ then
3:         Retrieves the set $N' \subseteq$N of child nodes of $n_i$
4:         Retrieves the set $Np \subseteq$N of child nodes of parent node of $n_i$
5:         $Np \leftarrow Np \cup N'$
6:         Remove $n_i$ of $Np$
7:     end if
8: end for
9: Return H
```

The method takes as input an entire processed hierarchy. Then it visits each node to check if the set of candidate labels is empty (line 2). If the set of candidate labels of a node n_i is empty, the existing connection with the node and its child nodes are removed and each child is linked to the parent node of n_i (lines 3-5). Finally, node n_i is removed from the hierarchy (line 6).

Particularly, the removal is important because SeCLAR will be used combined with a classical method to select labels as described in next Section. Many classical methods don't make any cutting in the hierarchy, and the final result after processing would be a cluster with multiple nodes without labels.

3.4 Combining SeCLAR with Traditional Methods

The traditional label selection methods presented in Section 2 receives as input a hierarchical clustering H, built from a set of documents D, a set of nodes N associated with H and a set of terms T previously selected from the set of terms present in the collection. They select the labels using as candidates all terms present in the set T.

The SeCLAR, as proposed in this work, selects the terms present in the set T, obtaining a subset $T' \subseteq T$ containing the terms considered as the best candidates labels. The SeCLAR receives as input a hierarchy H as described above, and the output is the hierarchy H' containing the set of documents D, the set $N' \subseteq N$ of nodes associated with the H', and the set T' of selected candidates terms.

Thus, a hierarchical clustering can be processed as demonstrated in Figure 1. For the approach using the traditional methods, the hierarchical clustering (step 1) is used as input for the traditional method (step 3). For the proposed approach, the hierarchy is sent to the SeCLAR (step 2). The output of this step is a hierarchy with the pre-selected candidates labels, which is sent to the traditional method (step 3). In both cases, the result is a hierarchical clustering with the selected labels (step 4).

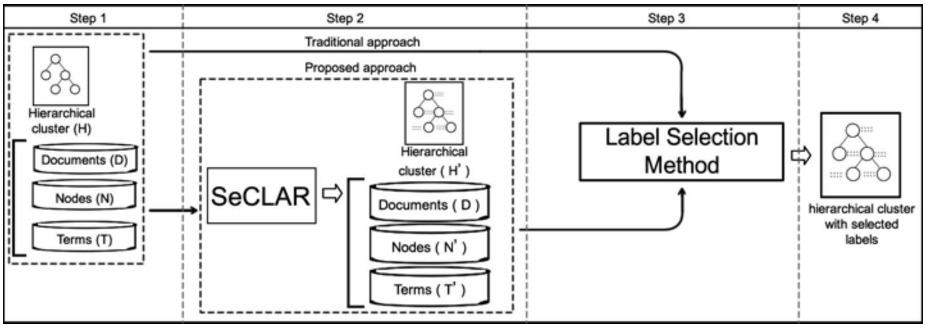


Fig. 1. Selecting labels using traditional approach and the proposed approach

The results obtained in the evaluations presented in Section 4 indicate that the combined use of SeCLAR and literature methods result in more specific labels and helps the recovery of documents organized by that hierarchy. This combination improves the level of analysis of traditional methods without modifying them.

4 Experiments and Results

There are no standard procedures to evaluate hierarchies although some attempts have been made. Besides the fact that the evaluation is difficult even when a group of volunteers is willing to participate. It also depends on the task the hierarchy is designed for [2]. The methodology proposed in [7] and described in Section 4.1 were used to compare the results obtained using the SeCLAR combined with the label selection methods presented in Section 2.

SeCLAR has some parameters that need adjustment, so the evaluation was divided into two stages. First, the different configurations for the SeCLAR are evaluated (Section 4.3). The second stage evaluates the impact of using SeCLAR along with methods present in the literature (Section 4.4).

For these evaluations 6 textual bases constructed from scientific documents such as articles and dissertations were used.

4.1 Evaluation Criteria

The evaluation process of the hierarchies was applied as proposed in [7], using a particular process of information retrieval. All label sets are used as search queries, connecting the terms with "*and*" operator. For example, if the set of labels of a node is *"{artificial,intelligence}"*, then the search expression will be *"artificial* $\wedge$ *intelligence"*. The final set of labels for each node are its own labels and the labels of all its direct ancestors to the root. For example, if the root has the set of labels *"{artificial,intelligence}"* and one of his sons has the set *"{data,mining}"*, then the search expression used to evaluate this child node is *"artificial* $\wedge$ *intelligence* $\wedge$ *data* $\wedge$ *mining"*.

A document associated with the node is recovered when all the terms of the search expression are present in this document. After the application of the retrieval process, for each cluster node, the values presented in Table 1 have to be calculated (t_p: documents correctly retrieved; f_p: incorrectly retrieved documents; f_n: documents not retrieved; t_n: documents correctly not retrieved; r_d: total number of documents retrieved; d_i: total number of documents that are associated to the node; x: total number of documents in the collection).

Table 1. Retrieval process results for each node

	$retrieved$	$!retrieved$	
n_i	t_p	f_n	$\lvert d_i \rvert$
$!n_i$	f_p	t_n	$x - \lvert d_i \rvert$
	r_d	$!r_d$	x

The precision and recall values for each node of the hierarchy are calculated according to the equations below:

– precision: the proportion of correctly retrieved documents among the total of retrieved documents ($prec = t_p/r_d$);
– recall: the proportion of the n_i documents that were retrieved ($rec = t_p/\lvert d_i \rvert$).

In order to evaluate the proposed method, a generalized linear model (for details see [11]) to analyse the variance and to obtain a mean estimate to each measure was adjusted. The model considers the effect of the node and the label selection method, as well as general average, as shown below:

$$\hat{m_e} = \hat{\mu} + \hat{n_i} + \hat{l_m} + \hat{e} \tag{1}$$

where:

- $\hat{m_e}$: evaluation measure estimate after the variance model adjustment;
- $\hat{\mu}$: general model mean, without any other effect;
- $\hat{n_i}$: the effect estimate of the node n_i, that is how much the $\hat{m_e}$ deviates from the $\hat{\mu}$ because of the node;
- $\hat{l_m}$: the effect estimate of the labeling method, that is how much the $\hat{m_e}$ deviates from the $\hat{\mu}$ because of the applied labeling method;
- $\hat{e}$: the random effect associated to each estimate.

To evaluate all the hierarchies in a similar way, it is considered only the nodes that are common to all evaluated clusterings. For this reason, a unique identity for each node is maintained.

According to [7], more statistically reliable comparisons are obtained using each measure estimate $\hat{m_e}$ for each labeling method $\hat{l_m}$. In this paper, the SNK test (Student-Newman-Keuls) are used for the analysis of variance and multiple comparison of means, with a 5% level of significance. The SNK test was chosen because of its strenght and its characteristic of always showing the real differences among tested means.

4.2 Hierarchies Preprocessing and Obtention

All text collections[1] were submitted to the same preprocessing method. Terms were obtained by reducing words to their stem using the Porter algorithm [9]. Then the terms were selected using the Salton filter [10], which suggests the use of terms that have the frequency of occurrence in the collection (DF - document frequency) between one and ten percent of the total documents. Since the documents are submitted to an agglomerative clustering process, the minimum value 2 was chosen for DF. Table 2 presents a description of the text collections. The initial amount of terms for each base refers to the stems obtained after applying the Porter algorithm and removing stopwords. The values of the last column indicate the number of stems after selecting attributes using Salton, i.e. the final cardinality of each text collection used for evaluation. In this work, only the generation of simple terms, also called *onegrams*, was considered.

Text collections were clustered using the average linkage algorithm and the cosine measure. Each node receives a unique identifier enabling to identify them even after the processing carried out by the label selection methods. These clusterings will be used in the evaluations in Sections 4.3 and 4.4.

[1] These text collections can be accessed from `www.icmc.usp.br/~fabianof/seclar`

Table 2. Details of textual bases used in this paper

Base	**# docs**	**# terms**	**Salton's DF filter**	**card(A)**
Computer Hardware	83	11159	$2 \leq DF \leq 10$	3076
A.I.	69	10225	$2 \leq DF \leq 9$	2614
Biophysics	72	13804	$2 \leq DF \leq 9$	3898
ifm-wp02	63	12126	$2 \leq DF \leq 8$	4854
ifm-wp04	47	14302	$2 \leq DF \leq 7$	5488
Inorganic Chemistry	74	18705	$2 \leq DF \leq 9$	5888

4.3 Evaluation of the SeCLAR Parameters

To evaluate the impact of the parameters of Algorithm 1 of SeCLAR the combinations of settings described in Table 3, totaling 48 combinations for each text collection were used. For all cases, the values of minimum support and minimum confidence were set at 10%. An adapted version of Apriori which generates association rules with one item in consequent was applied. This step consideres the 14 most frequent labels for each node. This was done so that each node had the same number of labels for evaluation. The value chosen in this work was defined in [7].

For each value of window size, 16 combinations were obtained and they were evaluated according to the methodology previously described.

The results of Tables 4 and 5 were evaluated, grouped according to the window size in order to maximize the number of nodes in common between the evaluated results. When evaluating all combinations at the same time, the number of nodes in common between the hierarchies tends to be very small, mostly corresponding to the leaf nodes. According to the criteria described in Section 4.1, the final average for each measure in each hierarchy would be formed only by the nodes that are not processed by SeCLAR, and therefore would not be possible to measure its impact.

After the experiments, it were observed that the parameter corresponding to the window size influences mostly the nodes removal step (presented in Section 3.3). This parameter determines the amount of transactions that will be obtained by

Table 3. Values used to evaluate the parameters of the proposed method

window_size	30	40	50	
K	10	50	100	200
measures	confidence (conf)	lift (lift)	laplace (lapl)	gini index (gini)
minsup and minconf	10%			

Table 4. Best results to precision

Precision						
	window size					
Base	30		40		50	
	configuration	$\hat{m_e}$ =prec	configuration	$\hat{m_e}$ =prec	configuration	$\hat{m_e}$ =prec
Computer Hardware	gini k=50	0,025253	lapl k=50	0,135626	conf k=200	0,027778
A.I.	lapl k=10	0,223886	lapl k=50	0,245405	lapl k=10	0,183361
Biophysics	lapl k=10	0,134401	lapl k=10	0,191065	lapl k=50	0,185637
ifm-wp02	lapl k=10	0,049123	lapl k=50	0,04533	lapl k=10	0,200101
ifm-wp04	lapl k=10	0,297434	lift k=200	0,260606	lapl k=10	0,295956
Inorganic Chemistry	lapl k=10	0,079458	lapl k=10	0,056525	lapl k=10	0,197809

Table 5. Best results to recall

Recall						
Base	window size					
	30		40		50	
Computer Hardware	configuration	$\hat{m_e}$ =rec	configuration	$\hat{m_e}$ =rec	configuration	$\hat{m_e}$ =rec
	lapl k=50	0,083802	lapl k=50	0,509933	lapl k=50	0,057485
A.I.	lapl k=10	0,759615	lapl k=50	0,799342	lapl k=10	0,779343
Biophysics	lapl k=10	0,497872	lapl k=10	0,661111	lapl k=50	0,712987
ifm-wp02	conf k=50	0,1438	lapl k=50	0,151884	lapl k=10	0,572995
ifm-wp04	lapl k=50	0,818966	lapl k=10	0,877273	lapl k=10	0,868103
Inorganic Chemistry	lapl k=10	0,239849	lapl k=10	0,204088	lapl k=10	0,694815

the transformation of the documents. Its value should not be too large because the number of transactions generated would be very small, which would make the use of association rules impractical. In this evaluation, the values 40 and 50 had better results when compared with the averages obtained using window size 30. Settings using the value 30 were better only in 3 cases, all for precision.

According to the results presented in Tables 4 and 5, the objective measure Laplace was dominant. It was better in 13 of 16 results for precision and in 15 of 16 results for the recall. The Laplace measure as used in this work, is a correction to the confidence measure. It applies a penalty to the calculation, giving preference to those rules that may represent the largest possible portion of the database. In this case, it is appropriate to the task of selecting labels, because it tries to find the subset of association rules that best describes the documents of each cluster. In particular, it reflects the hierarchical characteristic of clusters obtained, giving evidence that the relationships *parent-child* among nodes may be considered as a *antecedent-consequent* relations in the context of hierarchical cluster of documents.

The K parameter was mostly better for lower values (10 and 50). This indicates that even with a significant increase in the number of selected rules, only a small set of them contains the most interesting subset of terms to be used as labels. The values 10 and 50 were better in all results for the recall, and in 14 of 16 results for accuracy.

4.4 Evaluation of the Use of SeCLAR Combined with Traditional Methods

To evaluate the impact of using SeCLAR combined with traditional label selection methods, an experiment was carried out as described in this section. Clusterings obtained by the process described in Section 4.2 are processed as described in Section 3.4 and the details as given as follow: (i) Each clustering obtained was processed by the label selection methods Most Frequent, Popescul&Ungar and RLUM; (ii) Each clustering obtained was processed by SeCLAR, and then processed by Most Frequent, Popescul&Ungar and RLUM methods.

For methods that do not define a maximum value of labels per node, we considered the 14 most frequent labels for each one. This was done so that each

Table 6. Best settings for each evaluated database

	ifm1-wp02	**ifm1-wp04**	**Comp. Hard.**	**A.I.**	**Ino. Chem.**	**Biop.**
Measure	Laplace	Laplace	Laplace	Laplace	Laplace	Laplace
K	10	10	50	50	10	50
window_size	50	40	40	40	50	50

Table 7. Traditional approach vs Proposed approach

Base	Results								
	Method	precision				recall			
		configuration	n	$\hat{m}_e$ =prec	group	configuration	n	$\hat{m}_e$ =rec	group
Computer Hardware	MF	SeCLAR	147	0,085018	a	SeCLAR	147	0,181266	a
		Traditional	148	0,037162	b	Traditional	148	0,025115	b
	P&U	SeCLAR	147	0,085018	a	SeCLAR	147	0,181266	a
		Traditional	148	0,036036	b	Traditional	148	0,029443	b
	RLUM	SeCLAR	117	0,267990	a	SeCLAR	117	1,000000	a
		Traditional	117	0,262530	a	Traditional	117	0,974359	a
IA	MF	SeCLAR	119	0,104242	a	SeCLAR	119	0,206244	a
		Traditional	120	0,033333	b	Traditional	120	0,043938	b
	P&U	SeCLAR	119	0,086315	a	SeCLAR	119	0,182434	a
		Traditional	120	0,033333	b	Traditional	120	0,043938	b
	RLUM	Traditional	98	0,322250	a	SeCLAR	98	0,981293	a
		SeCLAR	98	0,299802	b	Traditional	98	0,981293	a
Biophysics	MF	SeCLAR	108	0,092196	a	SeCLAR	108	0,178858	a
		Traditional	109	0,045872	a	Traditional	109	0,031265	b
	P&U	SeCLAR	108	0,092196	a	SeCLAR	108	0,178858	a
		Traditional	109	0,045872	b	Traditional	109	0,031164	b
	RLUM	SeCLAR	82	0,255062	a	SeCLAR	82	1,000000	a
		Traditional	82	0,233793	a	Traditional	82	0,971545	a
ifm-wp02	MF	SeCLAR	125	0,100000	a	SeCLAR	125	0,036404	a
		Traditional	125	0,096000	a	Traditional	125	0,032404	a
	P&U	SeCLAR	92	0,184912	a	SeCLAR	92	0,451141	a
		Traditional	93	0,064516	b	Traditional	93	0,038526	b
	RLUM	SeCLAR	73	0,283447	a	SeCLAR	73	1,000000	a
		Traditional	73	0,262310	a	Traditional	73	0,956621	a
ifm-wp04	MF	Traditional	83	0.090361	a	SeCLAR	83	0.121600	a
		SeCLAR	83	0.062249	a	Traditional	83	0.051299	b
	P&U	Traditional	83	0.090361	a	SeCLAR	83	0.121600	a
		SeCLAR	83	0.062249	a	Traditional	83	0.051299	b
	RLUM	SeCLAR	66	0.341360	a	Traditional	66	1,000000	a
		Traditional	66	0.322601	a	SeCLAR	66	1,000000	a
Inorganic Chemistry	MF	SeCLAR	118	0.133660	a	SeCLAR	118	0.326762	a
		Traditional	119	0.046218	b	Traditional	119	0.033771	b
	P&U	SeCLAR	118	0.133660	a	SeCLAR	118	0.326762	a
		Traditional	119	0.046218	b	Traditional	119	0.033771	b
	RLUM	SeCLAR	94	0.285809	a	Traditional	94	1.000000	a
		Traditional	94	0.277041	a	SeCLAR	94	0.960993	a

node had the same number of labels for evaluation. The value chosen in this work was defined in [7].

SeCLAR parameters were selected from the analysis presented in Section 4.3, and the best settings for each base were selected (Table 6). These settings were obtained by observing the best results for recall, because usually the use of hierarchical organization of documents is related to problems of information retrieval.

We evaluated the results obtained without the use of SeCLAR to select the labels and combined version with SeCLAR, as presented in Section 3.4. The results are presented in Table 7.

The SeCLAR method contributed significantly to the improvement of the results obtained by Most Frequent and Popescul&Ungar methods. Considering precision, the SeCLAR improved statistically the average in eight cases, and did not obtain a statistical improvement in four cases (3 for MF and 1 for Popescul&Ungar). We believe that the results for the case of the Most Frequent method are related to its simplicity. Considering recall, there was a statistical difference in 11 of 12 cases. Therefore, the SeCLAR was able to significantly improve the coverage of these methods, without sacrificing the precision.

The behavior of the methods was exactly equal to the bases *ifm-wp04* and *Inorganic Chemistry*, i.e. the average of the Most Frequent method are identical to those of Popescul&Ungar. These methods usually present similar results [7]. A visual evaluation of the results indicated that due to the selection performed by SeCLAR, the behavior of both methods was identical in all nodes, thus generating a set of labels the same for both cases.

SeCLAR contributed to the increase of the precision value of the RLUM method, however it was not able to contribute significantly. In five cases there was no statistical difference. The first case presented a statistically better outcome for the use of RLUM without SeCLAR. This method naturally has good results for recall in this evaluation, often close to 1, and an improvement in this measure would be very difficult. The SeCLAR contributed to the increase of the average value for the measures, but these results aren't statistically significant.

5 Conclusions

This work proposes a new method for the selection of candidates labels for hierarchical clustering. The SeCLAR method explores more explicitly the direct relationships which exist in the hierarchical organization. The general idea of the proposed method is to consider each *parent-child* relationship between nodes as *antecedent-consequent* relations used in association rules. Moreover, it also explores a direct relationship between the terms of the documents, since it does not consider each document as a *bag-of-words*. In this proposal, the SeCLAR can be used regardless of the label selection method that would be chosen by adding features like the elimination of word repetition throughout the hierarchy.

The results obtained by the use of SeCLAR indicate that its use has contributed significantly to the selection of labels. In particular, the SeCLAR contributes to the improvement of precision in many cases, without reducing the coverage of the method (measured by recall). The SeCLAR method can also be used as a label selection method, and the results are close to those obtained by its combination with the Most Frequent method.

In future works, we intend to expand the evaluation to other objective measures of association rules. Furthermore we aim to evaluate the impact of SeCLAR in the selection of labels for hierarchical clustering with the support of domain experts. Finally, it would be interesting to investigate the possibility of combining the SeCLAR approach, explored in this paper, with other label selection methods present in the literature.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB 1994: Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499. Morgan Kaufmann Publishers Inc., San Francisco (1994)
2. Bast, H., Dupret, G., Majumdar, D., Piwowarski, B.: Discovering a term taxonomy from term similarities using principal component analysis. In: Ackermann, M., Berendt, B., Grobelnik, M., Hotho, A., Mladenič, D., Semeraro, G., Spiliopoulou, M., Stumme, G., Svátek, V., van Someren, M. (eds.) EWMF 2005 and KDO 2005. LNCS (LNAI), vol. 4289, pp. 103–120. Springer, Heidelberg (2006)
3. Glover, E.J., Pennock, D.M., Lawrence, S., Krovetz, R.: Inferring hierarchical descriptions. In: CIKM, pp. 507–514. ACM, New York (2002)
4. Larsen, B., Aone, C.: Fast and effective text mining using linear-time document clustering. In: KDD 1999: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 16–22. ACM, New York (1999)
5. Lopes, A., Pinho, R., Paulovich, F., Minghim, R.: Visual text mining using association rules. In: ScienceDirect, pp. 316–326 (2007)
6. Mahgoub, H., Rösner, D., Ismail, N., Torkey, F.: A text mining technique using association rules extraction. International Journal of Computational Intelligence, 21–28 (2008)
7. Moura, M.F., Rezende, S.O.: A simple method for labeling hierarchical document clusters. In: Proceedings of AIA 2010 - Artificial Intelligence and Applications, Innsbruck, Austria (2010)
8. Popescul, A., Ungar, L.: Automatic labeling of document clusters (2000) (unpublished manuscript), http://www.cis.upenn.edu/~popescul/Publications/popescul00labeling.pdf
9. Porter, M.F.: An algorithm for suffix stripping. Readings in Information Retrieval, 313–316 (1997)
10. Salton, G., Buckley, C.: Term weighting approaches in automatic text retrieval. Technical report, Ithaca, NY, USA (1987)
11. Searle, S.R.: Linear models. J. Wiley, New York (1971)
12. Treeratpituk, P., Callan, J.: Automatically labeling hierarchical clusters. In: dg.o 2006: Proceedings of the 2006 international conference on Digital government research, pp. 167–176. ACM, New York (2006)
13. Weiss, S.M., Indurkhya, N., Zhang, T., Damerau, F.J.: Text Mining - Predictive Methods for Analizing Unstructured Information. Springer Science+Business Media, Inc., Heidelberg (2005)

Recognizing Textual Entailment Using a Machine Learning Approach

Miguel Angel Ríos Gaona[1], Alexander Gelbukh[1], and Sivaji Bandyopadhyay[2]

[1] Center for Computing Research, National Polytechnic Institute, Mexico
mriosb08@sagitario.cic.ipn.mx, gelbukh@gelbukh.com
[2] Computer Science & Engineering Department, Jadavpur University, Kolkata 700 032 India
sivaji_cse_ju@yahoo.com

Abstract. We present our experiments on Recognizing Textual Entailment based on modeling the entailment relation as a classification problem. As features used to classify the entailment pairs we use a symmetric similarity measure and a non-symmetric similarity measure. Our system achieved an accuracy of 66% on the RTE-3 development dataset (with 10-fold cross validation) and accuracy of 63% on the RTE-3 test dataset.

Keywords: Recognizing Textual Entailment, text similarity measures, non-symmetric measures.

1 Introduction

One of the largest challenges in Natural Language Processing (NLP) is to provide a computer with the linguistic knowledge necessary to successfully perform language-oriented tasks. For example, for the query "*What does Peugeot manufacture?*" a Question Answering (QA) system must be able to recognize, or infer, and answer which may be expressed differently from the query. For example, from a text "*Chrétien visited Peugeot's newly renovated car factory*" the system should be able to infer a hypothesized answer from "*Peugeot manufactures cars*". A fundamental phenomenon in NLP is the variability of a semantic expression: the same meaning can be expressed in, or inferred from, different text.

A task that addresses this inference phenomenon is Recognizing Textual Entailment (RTE). Textual Entailment is defined as a directed relationship between pairs of text expressions, denoted by T (text) and H (hypothesis). We say that T entails H if the meaning of H can be inferred from the meaning of T as could typically be interpreted by people [3].

Moreover, many NLP tasks have strong relationship to entailment: in summarization, a summary should be entailed by the text; paraphrases can be seen as mutual entailment between a text T and a hypothesis H; in Information Extraction (IE), the extracted information should also be entailed by the text; in Question Answering (QA) and Information Retrieval (IR), the answer obtained for a query must be entailed by the supporting snippet of text.

To address the RTE task, different methods have been proposed, with varying degree of success. These methods can be classified by the type of representation of the entailment pair. The commonly used criteria for entailment recognition are similarity measures between T and H, the coverage of H by T in lexical representation methods and lexical-syntactic representation methods, and the ability to infer H from T, in the logical representation approach. Some authors [8] try to detect non-entailment, by looking for various kinds of mismatch between the text and the hypothesis.

In this paper, we propose the use of a symmetric similarity measure and a non-symmetric similarity measure as features in a Machine Learning (ML) algorithm for RTE. The symmetric measure is the cosine string similarity measure. The non-symmetric measure is given by measuring the causal relation between the entailment pairs. This measure uses the relative frequencies of words in a cause-effect set. The cause-effect set is created by retrieving sentences from the Web that contain the discourse marker *because*.

The hypothesis behind our system is that the symmetric similarity measures can not answer correctly (cover) all the entailment pairs, and with the addition of the non-symmetric similarity measures the remaining pairs might be covered.

The paper is structured as follows. In Section 2, we present an overview of related work. In Section 3, we describe the measures used in our experiments. In Section 4, we give the experimental results and the comparison with the state of the art. Finally, Section 5 concludes the paper.

2 Related Work

The RTE approaches can be classified by the textual entailment phenomena they address or by type of linguistic representation (*levels of language*) of the entailment pair they use. Each type of linguistic representation requires its own operations in order to establish the entailment decision, e.g., word matching at the lexical level, tree edit distance at the syntactic level, etc.

In some systems, the entailment decision ("T entails H") is made by comparing the score of the given operation with a threshold learned from an annotated corpus: If this score is greater than the threshold, the system answers "true", otherwise the answer is "false". There are different techniques to learn a threshold.

The main operations on a linguistic representation are similarity measures. Most of these similarity measures are symmetric. However, a symmetric measure can not capture important aspects in the $T \rightarrow H$ (T implies H) relation. For example, if we alter the entailment relation (i.e., $H \rightarrow T$) a symmetric function will give the same score. Therefore, some authors, e.g., [14], propose a non-symmetric similarity measure. Such measures have been used in RTE-1 Challenge.

Glickman [5] defines the entailment relation as follows: T entails H if $P(H \mid T) > P(H)$. The probabilities are calculated on the base of the Web. The accuracy of this system is the best for RTE-1 (56%).

Another non-symmetric method was proposed by Kouylekov [9], who uses the definition: T entails H if there exists a sequence of transformations applied to T such that H is obtained, with a total cost below of a certain threshold. The following transformations are allowed: insertion: insert a node from the dependency tree of H into the dependency tree of T; deletion: delete a node from the dependency tree of T; substitution: change a node in the T for a node of H. Each transformation has a cost and the cost of edit distance between T and H, ed(T, H) is the sum of costs of all applied transformations. The entailment score of a given pair is calculated as

$$\text{score}(T, H) = \text{ed}(T, H),$$

If this score is below a learned threshold, the relation T → H holds. The accuracy of this method is also of 56%.

In [14], an even "more non-symmetric" measure is proposed: when the edit distance (which was a modified Levenshtein distance) fulfills the relation:

$$\text{ed}(T,H) < \text{ed}(H,T),$$

then the relation T → H holds.

Other authors use a definition that in terms of representation of knowledge as feature structures could be formulated as: T entails H if H subsumes T [14]. The method used in [3] is also non-symmetric: T entails H if H is not informative in respect to T.

A method of establishing the entailment relation could be obtained using a non-symmetric measure of similarity between two texts presented by Corley and Mihalcea [2], who define the similarity between the texts T_i and T_j with respect to T_i as:

$$sim(T_i, T_j)_{Ti} = \frac{\sum_{pos}\left(\sum_{wk \in ws^{ti}_{pos}} (\max Sim(w_k) \times idf(w_k))\right)}{\sum_{pos}\sum_{wk \in ws^{ti}_{pos}} idf(w_k)}$$

Here the sets of open-class words (nouns, verbs, adjective, and adverbs) in each text segment are denoted by the PoS (part of speech) of WST_i and the PoS of WST_j. For a word w_k with a given PoS in T_i, the highest similarity of the words with the same PoS in the other text T_j is denoted by maxSim(w_k).

Basing on this text-to-text similarity metric, we derive a textual entailment recognition system by applying the lexical refutation theory [14]. As the hypothesis H is less informative than the text T, for a TRUE pair the following relation will hold:

$$\text{sim}(T, H) \times T < \text{sim}(T, H) \times H.$$

This relation can be proved using lexical refutation. A general scheme of the solution is the follows: to prove T → H it is necessary to prove that the set of formulas {T; neg-H} is lexically contradictory (T and negH also denote the sets of disjunctive clauses of T and negH).

3 Similarity Measures Used in the Experiments

Many systems for RTE are based on similarity measures; we used these measures to train a machine learning algorithm. The entailment decision is given by a classifier, where the classes are "true" and "false".

We will now describe the two string similarity measures used in our experiments. We chose two measures as features: the cosine symmetric measure and the causal non-symmetric measure.

3.1 Cosine Similarity Measure

Large classes of measures of semantic similarity are best conceptualized as measures of vector similarity. We consider binary vectors, that is, vectors with entries that are either 0 or 1. The simplest way to describe a binary vector is as the set of its nonzero values.

Cosine similarity is a measure of similarity between two *n*-dimensional vectors obtained by finding the cosine of the angle between them. It is often used to compare documents in text mining. In addition, it is used to measure cohesion within clusters in data mining. Cosine similarity is also widely used in information retrieval to calculate the similarity between documents or sentences. Given two vectors of attributes, *A* and *B*, the cosine similarity θ is calculated using the dot product and magnitude as:

$$COS(A,B) = \frac{|A \cap B|}{\sqrt{|A| \times |B|}} .$$

Note that this is a symmetric measure, that is, COS(*A*, *B*) = COS(*B*, *A*).

3.2 Causal Non-symmetric Measure

First, we give a brief theoretical introduction to the measure. A causal relation refers to the relation between a cause and its effect or between regularly correlated events. The type of coherence relation we used is cause-effect is illustrated below. In the example below, (1) states the cause for the effect given in (2):

1. *There was bad weather at the airport*
2. *Our flight was delayed.*

The causal relation subsumes the cause and the explanation relations discussed by Hobbs [7]. Hobbs's causal relation holds if a discourse segment stating a cause occurs before a discourse segment stating an effect; an explanation relation holds if a discourse segment stating an effect occurs before a discourse segment stating a cause. The causal relation is encoded by adding a direction. In a graph, this can be represented by a directed arc going from cause to effect.

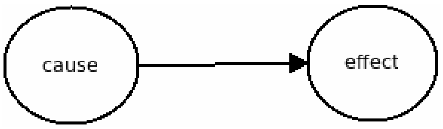


Fig. 1. Cause-effect graph

In Figure 1, the causality is a directional relationship, in the same way as the relationship between the members of an entailment pair. A non-symmetric similarity measure based on the count of co-occurrences of causal lexical pairs could be as follows: If a word *x* is a necessary (likely) cause of a word *y*, then the presence of *y* necessarily (likely) implies the presence of *x*.

In [11], a non-symmetric similarity measure is proposed based on the treatment the entailment pair as a causal relation, where the text T is a cause and the hypothesis H is its effect, i.e., T causes H. The non-symmetric similarity measure is based on the count of co-occurrences of causal lexical pairs from cause-effect (C-E) pairs extracted from a corpus.

Algorithm 1. The non-symmetric similarity measure

```
For each word $t_i$ in $T$
    For each word $h_j$ in $H$
        $ce_j$ = causal frequency($t_i$,$h_j$)
        $e_j$ = causal frequency($h_j$)
    $max_i$ = argmax($ce_j$ / $e_j$)
nonsymmetric($T$,$H$) = $\sum max_i$
```

In Algorithm 1 used for our experiments, the first causal frequency function is the count of words t_i and h_i related by a cue phrase (for example, a sentence "H … *because* … T") in a corpus of C-E pairs, and the second causal frequency function is the count of word h_i in the C-E pairs. This gives a non-symmetric score, because the frequency counts of "T causes H" is not the same as "H causes T".

4 Experimental Results

In this section we first describe the linguistic processing for feature extraction and then the experiments over various Machine Learning algorithms. Finally, we give a comparison with the state of the art.

4.1 Experimental Setting

The linguistic processing we used with each entailment pair is as follows:

1. *Tokenizing*. As usually, the first step of processing is to divide the input text into units called tokens. Each of them is a *word,* a number, a punctuation mark, etc. The treatment of punctuation marks can vary in such process. Our system just strips the punctuation marks out. We consider as word any string between whitespaces and punctuation characters. The whitespace is the main clue used in English texts (RTE benchmark is in English).

2. *Removal of stop words*. The system removes any stops words that are listed in the corresponding list, such as *the*, *from*, or *could*. These words have important semantic function in English, but they rarely contribute information if the criterion is a simple word-by-word match.

3. *Measuring similarity*. Similarity measures are applied to each entailment pair, to extract the train and test sets for the machine learning algorithm.

The data we used to collect the frequency of the causal lexical pairs for the causal non-symmetric measure was from training sentences which contain the cue word *because*. The causal sentences were separated in two parts: one corresponding to the cause and the other one corresponding to its effect, to finally form the cause-effect pairs. The sentences were extracted from the Sketch Engine system over a large corpus (ukWAC from the Sketch Engine[1]). The Sketch Engine is a corpus query system that allows the user to view word sketches, thesaurally similar words, and so-called "sketch differences", similarly to the usual Corpus Query Systems (CQS).

4.2 Machine Learning Experiments

The RTE-3 Challenge provided two datasets (a development dataset and a test dataset), each one consisting of 800 entailment pairs. In both datasets, pairs are annotated according to the task. In RTE-2 the length annotation is introduced, with values of either "long" or "short." In addition, the development sets are annotated as to whether each pair is in the entailment relation or not.

We applied the linguistic preprocessing to each RTE-3 dataset; the result is a set of vectors of two features. These sets are used to train and test a classifier. We used the WEKA[2] machine learning platform [15] for our experiments.

We ran several experiments with various machine learning algorithms, including Support Vector Machine, AdaBoost, Naïve Bayes, among others. We used the RTE-3 development dataset to train the classifiers. The results of the 10 fold-cross validation are show in Table 1.

The Support Vector Machine (SVM) and the Naïve Bayes achieved the best results in the experiments during the training phase. Then we used these two algorithms to perform the classification over the RTE-3 test dataset.

Table 1. 10 fold-cross validation results over the RTE-3 development dataset

Algorithm	Accuracy
SVM	**66.37%**
NaïveBayes	**65.87%**
AdaBoost	65.25%
BayesNet	65.25%
LogitBoost	65%
MultiBoostAB	64.125%
RBFNetwork	64.87%
VotedPerceptron	51.75%

The SVM algorithm tries to compute the hyperplane that best separates the set of training examples (the hyperplane with maximum margin). On the other hand, the Naïve Bayes algorithm is a classification algorithm based on the Bayes rule that assumes the features are all conditionally independent from one another. The value of this assumption is that it dramatically simplifies the representation of the probability P(X | Y) and the problem of estimating it from the training data.

[1] http://www.sketchengine.co.uk/

[2] WEKA. http://www.cs.waikato.ac.nz/ml/weka/

4.3 Comparison with Previous Results

The experimental results are summarized in Table 2. We compare our system against other Machine Learning systems which use features based on similarity measures. All the systems were tested over the RTE-3 test dataset. In Table 2 we report the results with the Naïve Bayes. The SVM achieved an accuracy of 62.87%.

Table 2. Comparison with previous results

System	Number of Features	Accuracy
Our system with Naïve Bayes	**2**	**63.5%**
Li et al. (2007)	7	62.75%
Malakasiotis and Androutsopoulos (2007)	10	61.75%
Ferrés and Rodriguez (2007)	12	61.50%

Therefore, our system outperformed the other machine learning systems, which used more features. Indeed, we used only two features (one symmetric and one non-symmetric similarity measure), while, for example, in [11] the authors used 10 different similarity measures (e.g. Levenshtein distance, Jaro-Winkler, Soundex, etc.).

The approach of Ferrés and Rodriguez [5] for computing distance measures between sentences is based on the degree of overlapping between the semantic content of the two sentences. Obtaining the semantic content implies deep linguistic processing. Upon this semantic representation of the sentences, several distance measures are computed.

Li *et al.* [10] produced seven features for each entailment pair: lexical semantic similarity, named entities, dependent content word pairs, average distance, negation, task, and text length. The last two features are extracted from each pair itself, while others are based on the results of language analyzers.

Finally, the system of Malakasiotis and Androutsopoulos [11] uses SVM's to determine whether each T–H pair constitutes a correct textual entailment pair. In particular, it employs four SVMs, each trained on the development dataset of the corresponding RTE subtask (QA, IR, IE, SUM) and used on the corresponding test dataset. Preliminary experiments indicated that training a single SVM on all four subsets leads to worse results, despite the increased size of the training set, presumably because of differences in how the pairs were constructed in each subtask, which do not allow a single SVM to generalize well over all four. Their system is based on the assumption that string similarity at the lexical and shallow syntactic level can be used to identify textual entailment.

Thus, many ML systems need a complex linguistic processing in order to extract features for modeling the entailment recognition.

5 Conclusions and Future Work

We proposed combined use of symmetric similarity measure and non-symmetric similarity measure as features for a machine learning approach. We have shown that our system outperforms other machine learning approaches to RTE. Furthermore, we have shown that the use of two different types of measures improves the performance of a machine learning system. Finally, our system has the advantage of simplicity and the use of a very limited feature set.

Our system also has competitive accuracy, because the average accuracy for the RTE-3 is about 61%. The state-of the-art (shown by non-machine learning-based systems) for the RTE-3 is about 80%.

In the future we plan to use other non-symmetric similarity measures, i.e., Corley and Mihalcea, Glickman. We will use syntactic and semantic measures (WordNet-based similarity measures) to achieve better performance. In particular, we plan to test deeper semantic processing, including determining and using verb valencies [1]. Finally, we will test our system over the past RTE Challenge datasets as new test and training sets.

Acknowledgements. The work was done under partial support of Mexican Government (SNI, CONACYT grant 50206-H, CONACYT scholarship for Sabbatical stay at Waseda U., COFAA-IPN, and SIP-IPN grant 20100773) and Government of India (CONACYT-DST India funded project).

References

1. Castro-Sánchez, N.A., Sidorov, G.: Analysis of Definitions of Verbs in an Explanatory Dictionary for Automatic Extraction of Actants based on Detection of Patterns. In: Hopfe, C.J., Rezgui, Y., Métais, E., Preece, A., Li, H. (eds.) NLDB 2010. LNCS, vol. 6177, pp. 233–239. Springer, Heidelberg (2010)
2. Corley, C., Mihalcea, R.: Measuring the semantic similarity of texts. In: Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, Ann Arbor (2005)
3. Dagan, I., Glickman, O.: Probabilistic textual entailment: Generic applied modeling of language variability. In: PASCAL workshop on Text Understanding and Mining (2004); Monz, C., de Rijke, M.: Light-Weight Entailment Checking for Computational Semantic. In: Blackburn, P., Kohlhase, M. (eds.) Proceedings ICoS-3 (2001)
4. De Salvo Braz, R., Girju, R., Punyakanok, V., Frentiu, D.M.: An Inference Model for Word Sense Disambiguation. In: Proceedings of KEPT 2007, Knowledge Engineering Principles and Techniques, Workshop on Recognising Textual Entailment, vol. I (2007)
5. Ferrés, D., Rodríguez, H.: Machine Learning with Semantic-Based Dis-tances Between Sentences for Textual Entailment. In: Proceedings of the Third Challenge Workshop Recognising Textual Entailment, Prague, Czech Republic (2007)
6. Glickman, O., Dagan, I., Koppel, M.: Web Based Probabilistic Textual Entailment. In: Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment (2005)
7. Hobbs, J.R.: Ontological promiscuity. In: Proceedings of the 23rd annual meeting on Association for Computational Linguistics (1985)
8. Inkpen, D., Kipp, D., Nastase, V.: Machine Learning Experiments for Textual Entailment. In: Proceedings of the Second Challenge Workshop Recognising Textual Entailment, Venice, Italy, April 10, pp. 17–20 (2006)
9. Kouylekov, M., Magnini, B.: Tree Edit Distance for Recognizing Textual Entailment: Estimating the Cost of Insertion. In: Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment, Venice, Italy (2006)

10. Li, B., Irwin, J., Garcia, E.V., Ram, A.: Machine Learning Based Semantic Inference: Experiments and Observations at RTE-3. In: Proceedings of the Third Challenge Workshop Recognising Textual Entailment, Prague, Czech Republic (2007)
11. Malakasiotis, P., Androutsopoulos, I.: Learning Textual Entailment using SVMs and String Similarity Measures. In: Proceedings of the Third Challenge Workshop Recognising Textual Entailment, Prague, Czech Republic (2007)
12. Pérez, D., Alfonseca, E.: Application of the Bleu algorithm for recognising textual entailments. In: Proceedings of the First Challenge Workshop Recognising Textual Etailment, Southampton, U.K (2005)
13. Ríos, M., Gelbukh, A., Bandyopadhyay, S.: Recognizing Textual Entailment with Statistical Methods. In: MCPR 2010, 2nd Mexican Conference on Pattern Recognition (2010) (to be published)
14. Tatar, D., Gabriela, S., Andreea-Diana, M., Rada, M.: Textual Entailment as a Directional Relation. Journal of Research and Practice in Information Technology (2009)
15. Witten, H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco (2005)

Detection of Different Authorship of Text Sequences through Self-organizing Maps and Mutual Information Function

Antonio Neme[1], Blanca Lugo[2], and Alejandra Cervera[3]

[1] Complex systems and nonlinear dynamics group,
Universidad Autónoma de la Ciudad de México, México, D.F. México
`antonioneme@gmail.com`
`http://www.nolineal.org.mx`
[2] Facultad de Ciencias, Universidad Autónoma del Estado de Coahuila,
Saltillo, Coah, México
[3] Comisión Nacional para el uso y conocimiento de la biodiversidad, México

Abstract. Writers tend to express their ideas with different styles, defined with the so called firm or stylome, which is an abstraction of the general constraints and specific combinations of words within their language they decide to follow. Although capturing this style has proven to be very difficult, some advances have been achieved. Here, we present a novel system that is trained with texts from the same author, and is able to unveil some of its features, and to apply them to detect texts not written by the same author, or, at least, not written with the previously learned features. The system is an hybrid model based in self-organizing maps and in information-theoretic aspects. In the model, mutual information function of unknown texts are compared to the mutual information function of texts from a known author. If the distance between these two distributions exceeds a certain threshold, then the unknown text is from a different author, otherwise the authorship is the same. The decision threshold is obtained by the self-organizing map trained with the texts from the same author. We present results in authorship identification in several contexts including classic literature, journalism (political, economical, sports), and scientific divulgation.

Keywords: authorship identification; anomalies detection; mutual information; self-organizing maps.

1 Introduction

Authorship identification is a relevant task. In this problem, a text whose author is not identified has to be compared in some way with other texts in order to identify its authorship, or, at least, to rule out some possible authors. A human expert is able to discriminate the authorship of a given text, within certain constraints. That is, if he/she is presented with a text whose author is not labeled, she may identify the name of the author, as an expert, she is supossed to have learnt the style and structural relationships authors tend to follow in their texts. A particular case of this task is to decide if the author of a text is a given individual or not [1].

The latter task is equivalent to that of anomaly detection, in which a system is presented with a set of normal or habitual behavior [2]. When another behavior is presented to the system, it has to decide if it is normal or abnormal, that is, if it is similar to the previous behaviors. This system should be able to learn the correlations in those behaviors, such that the relevant ones are able to discriminate the abnormal from the normal states [3].

A system able to decide if a text has as its author a given writer has to be trained with some texts from that author. From those texts, the system has to elucidate a group of measures, or identify some regularities, that are present in other texts from the same author, but not in texts whose authorship is different [4]. In this sense, it is a particular case of one-class learning, in which several positive examples (member of class) are presented to the system, but no instance of a non-member example is presented [5].

Several methods have been proposed, based in vocabulary size, syntactic patterns, length of sentences, among others. There are many authorship attribution methods based on soft-computing [6], several contributions based on computational linguistics [7], and many others based in statistical methods [8,9].

2 The Model

In our model, neither the lexicon nor the length of sentences, nor any other syntactical structure is taken into account. We process the text as a time series, and study them with non-linear time series analysis tools. The tools for time series analysis we apply are mutual information function and self-organizing maps.

Our proposal is summarized as follows. A number of texts from the same author define the habitual or common set. This set is the base for the method to learn and identify the features that describe the stylistics of the author, and that are supposed to be unique. Each one of this texts is processed as a time series and the mutual information function is obtained for each one. Then, the average mutual information function for all texts from the habitual author is calculated. At the same time, all texts from the habitual set are mapped through a one-dimensional self-organizing map, from where we obtain the weightened center of mass of each text. The average distance between centers of mass is calculated and from this quantity, we obtain a threshold. A text from a possibly different author is presented to the system, so it obtains the mutual information function of it, and compares this function to the average mutual information function previously achieved from texts from the known or base author. If the distance between these two functions exceeds the threshold indicated by the self-organizing map, then the new text is identified as a text written by a different author.

It models the text from an author as a time series, in which time is only the position of a word in the text, an the state associated to a word is its order of apparition. The first word in the text is assigned to state 0, so the number of

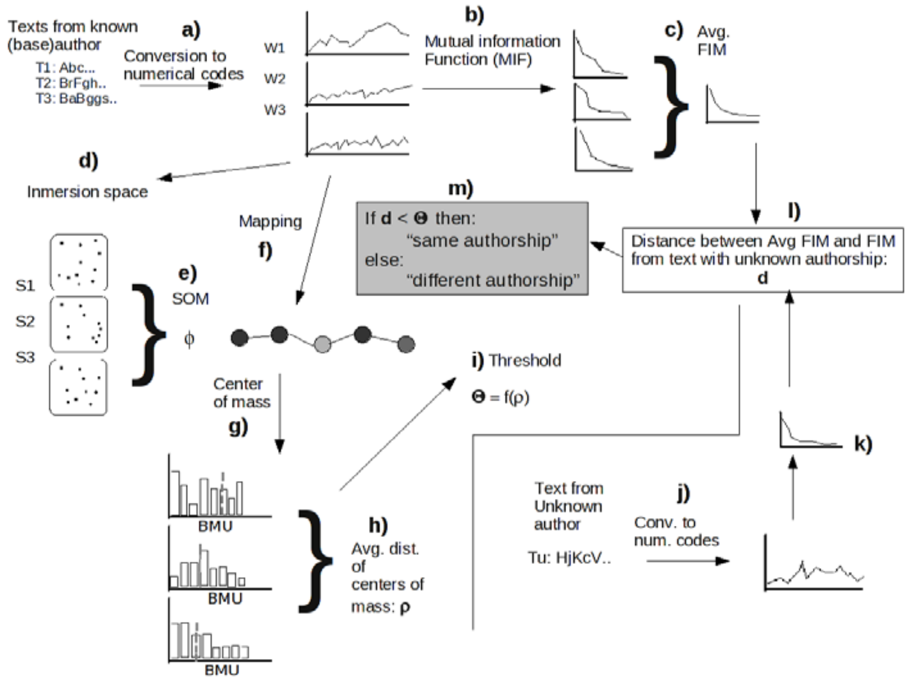


Fig. 1. Visual description of the proposed model for authorship identification

different symbols or states S, is 1, and the symbol or state sequence $Q = [0]$. From the second word, if it has not previously appeared, then its state is assigned to the number of different states, followed by an increment of one in S. If the current word i has previously appeared as a previously unseen symbol w, then it is assigned that position. Formally, we have a transformation of a text into a series of integer numbers, or symbols. W is the list of words to be transformed, and $W[n]$ refers to word in position n in list W. The new list of states or symbols is Q. The transformation process is stated by the following algorithm:

```
1.The first word, at position $n = 1$ in $W$, is assigned state 0, so $Q = [0]$.
    The number of different states (words) is $S = 1$.
    The next word to analyze is in position $n = 2$.
 2. Repeat until all words in the text have been transformed.
       $p = h(W[n], Q)$ # verify if word W[n] has previously appeared
 3.    if $p = -1$: # word has not previously appeared.
           $Q[n] = S$
           $S = S + 1$ # Increment the number of different words.
 4.    else:
           # word has previously appeared and state
           # p has assigned to the first appeareance of it.
           $Q[n] = p$
```

where the function $h(a, Q)$ returns the position of symbol a in the list Q, or -1 if symbol a is not in Q. Q is the same lenghth as W, but now we have a numeric list, which may be studied as a time series. From the new list W, me may apply time series analysis. The first obvious step is to discretize this time series in M codewords or symbols. In a given text, and varying from author to author, there could exist up to thousands of different words or symbols, so a discretization would reduce computing time. The range of different symbols $[1, S]$ is reduced to a discrete range in $1, 2, ..., M$ through a linear projection $x \in [1, S] \rightarrow int(x \times M/S)$.

One tool frequently applied in the nonlinear time series analysis context is mutual information function, or MIF. It is widely applied to seek and quantitatively characterize correlations between data that are not detected by linear measures of correlation [14]. Mutual information function is able to unveil nonlinear correlations between system X and system Y. It is a measure of information between two possibly nonliearly correlated systems. It has been applied as an abstraction of several sequences, such as in the context of molecular biology, in which it is presented as an alternative form of the so called genomic signature [15]. MIF has been able to generate very short descriptions of genomes at the time that it approximates a bijective function between strings over an alphabet of four symbols and quasi-unique descriptions [16]. It has been stated that MIF from related organisms tend to be similar, whereas MIF from phylogenetically distant organisms tend to be very different [16].

From the succesful results of MIF in the molecular biology context, we propose its use in the authorship identification problem. In order to determine the authorship of a text, we propose the use of MIF as a first step. In the application of MIF in time series analysis, there are not two systems, but only one. Thus, system Y is to be created from system X by a shift of k positions. That is, if $X = [0, 1, 2, 3, 4, 0, 5, 6, 2, ..., iN - 2, N - 1, N]$, a shift $k = 1$ leads to system $Y = [1, 2, 3, 4, 0, 5, 6, 2, ..., N - 1]$ and a shift of $k = 2$ will lead to $Y = [2, 3, 4, 0, 5, 6, 2, N - 2]$. The second system is thus the first one with a displacement of k positions. MIF answers the question if I am looking at symbol i, how much information does it give about the symbol located k positions downstream?

MIF is then a vector with K components in which each component k represents the average information symbol i gives about the symbol j located k positions downstream from symbol i. From this, we can calculate the distance between two MIF, as each MIF is a point in a $K-$dimensional space. K is the maximum displacement for whom MIF is calculated.

We propose that MIF is a good method to capture the so called stylistics or stylome of an author. In fig. 2, it is presented the MIF from three texts from the same author, and MIF for a text from a different author. It is observed that the texts from the same author present the similar MIF, whereas texts from different authors presents differences in MIF.

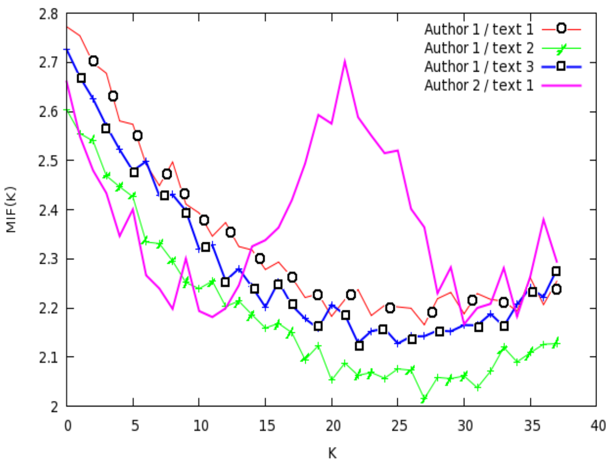


Fig. 2. Mutual information function for three texts from the same author and for a text from a different author

The use of MIF as an authorship identification is as follows:

```
1. For each text $i$ of a given author $a$, obtain $MIF(i, K)$.
   Calculate the average MIF, $MIF_a(K)$.
2. For the text $j$ of an unknown author, calculate $MIF(j, K)$.
   Obtain distance between the two MIF, $d_{ij} = \text{dist}(MIF_a(K), MIF_j, K))$.
3. If $d_{ij} \geq \Theta$:
       authorship of text $j$ does not correspond to author $a$
4. else:
       text $j$ has the same authorship $(a)$.
```

Obtaining a threshold Θ such that the authorship identification process shows as low errors as possible is a critical task. We propose that the non-supervised neural network called self-organizing map constitute a good method to calculate Θ. The self-organizing map (SOM) is a model of neural connections that is able to achieve organization from disordered configurations [10]. One of the major properties of the SOM is the ability to preserve in the output map those topographical relations present in the input data, a highly desirable property for data visualization and clustering [10,12]. This attribute is achieved through the transformation of an incoming signal pattern of arbitrary dimension into a low-dimensional discrete map (usually one or two-dimensional) and by adaptively transforming data in a topologically ordered fashion [10,11]. Each input data is mapped to a single neuron in the lattice, to the one with the closest weight vector to the input vector, or best matching unit (BMU). The SOM preserves neighborhood relationships during training through the learning equation, which establishes the effect that each BMU has over any other neuron.

The SOM structure, or output map, consists of a low-dimensional lattice of homogeneous units. Each unit n maintains a dynamic weight vector w_n which is the basic structure for the algorithm to lead to map formation. The dimension of the input space is considered in the SOM by allowing weight vectors to have as many components as features in the input space. Weight vectors are adapted accordingly to:

$$w_n(t+1) = w_n(t) + \alpha_n(t)h_n(g,t)(x_i - w_n(t)) \tag{1}$$

where $\alpha(t)$ is the learning rate at epoch t, $h_n(g,t)$ is the neighborhood function from BMU neuron g to neuron n at epoch t and x_i is the input vector. In general, the neighborhood decreases monotonically as a function of the distance from neuron g to neuron n, and as a function of time. The SOM preserves relationships in the input data by starting with a large neighborhood and reducing it during the course of training [10]. Both, neighborhood and learning parameter are reduced by an annealing scheme.

SOM is able to learn some features that distinguishes the texts from the same author. Here, we applied a one-dimensional SOM, that is, a chain of units to map $p-$dimensional input vectors.

We propose the use of SOM as follows. Let T_A be the group of texts from author A, and T_A^i be the ith text from author A. From the first algorithm, each text is a time series of integers, which allows to construct vectors embedded in a $p-$dimensional space. The $t-th$ integer of the time series is associated to a vector $v(t)$ constructed as a window of length p:

$$v(t) = [x(t), x(t-1), ..., x(t-p+1)]^T \tag{2}$$

Once each text, represented as a time series, has been associated to a set of vectors, a low-dimensional representation of the distribution of those vectors is formed (see. fig. 1). That representation is achieved through SOM, which intends to approximate the density of vectors in the p-dimensional space. Each vector is mapped to an unit, known as the best matching unit or BMU. Vectors that are located in close positions in the $p-$dimensional space tend to be mapped to close BMU, whereas vectors located in distant regions in the multidimensional space tend to me mapped to distant BMU. This low-dimensional representation is the base for the threshold Θ that distinguishes the authorship of texts, and now, we explain why. For each text of the same author A, T_A^i, the center of mass is calculated, on basis of the number of vectors mapped to each unit. The distance between pairs of centers of mass is a measure how different are two texts from the same author. Each pair of distance between centers of mass is relevant to compute the typical distance between texts from a given author, in the map space. Once trained, all vectors are mapped to the SOM and the BMUs for each vector is annotated. The center of mass of text t is calculated from the list of BMUs:

$$CM_t = \sum_{i=1}^{N} i \times p_i \tag{3}$$

where p_i is the relative frequency by which unit i was selected as BMU for vectors from text t. The average distance between centers of mass for all available texts from the same author is calculated. This quantity is CM. Θ, the threshold, for deciding if two MIF come from the same author, is a function of CM. That is, it is necesary an equivalence between how far the texts from the same author were mapped in the SOM, and the maximum distance that separates their MIF. This maximum distance is the decision threshold.

As there are K displacements in the MIF, and there is a chain of N units in SOM, an equivalence has to be calculated. That is, if two data sets are instances of the same process (in this case, are texts from the same author) such that their prototypical units are separated by MD units (distance MD), then the distance between MIF for the two data sets should not exceed the threshold given by equation 4.

There are two constraints for this function: 1. $\Theta = f(0) = 0$, and 2. $\Theta = f(N) = \alpha^* \times K$. The first constraint states that if the average distance for the centers of mass is 0, then, a threshold higher than 0 will lead to the decision of another authorship of the text. The second one states that if the average distance of centers of mass from texts of the same author is as high as possible (N), then the threshold is maximum, that is, all possible texts will be considered as coming from the same author. Of course these are the two extremes possibilities.

Threshold Θ may be any function (linear, quadratic, logarithmic, exponenially), as long as if $r = 0$, $\Theta = 0$, and if $r = N$, $Theta = \alpha^* \times K$. We found out that the best function in means of ROC is:

$$\Theta = (N^2 - (CM - N)^2)^{\frac{1}{2}} \times \alpha^* \times K/N \tag{4}$$

where N is the number of units in SOM, CM is the average distance between the center of mass of the texts from habitual author, K is the maximum number of shifts in MIF, and α^* is the maximum mutual information found for any shift $\leq K$.

In order to use SOM, at least two texts from an author have to be presented to the SOM, otherwise, there will not be a direct way to calculate Θ. Some alternatives have been proposed to estimate it, such as that in [13], but here we suppose there are at least two texts from each author to calculate the decision threshold.

By using this hybrid system, we are able to estimate the temporal relation in texts through the MIF and also, we are able to estimate the spatial distribution of symbols or words in the space of sequences of symbols.

MD per se is not a good authorship discriminat as it is achieved from a static distribution, that is, vectors from texts are only a representation which do not consider time [13].

As was explained in the introduction, the process of authorship identification is related to the anomaly detection problem, in which a system is presented with some examples of normal behavior. When a new behavior is presented to the system, it has to decide if it is a normal behavior or if it is an anomaly or novelty. This is an instance of one-class classification, and it is an open problem in the machine learning and time series processing areas. In our contribution, the texts from one author defines the habitual behavior, to be compared to other texts in order to decide if they come from the same or other author. The proposed model is not based in any vocabulary measure, but in the relation followed by the use of vocabulary.

3 Results

The hybrid system of MIF and SOM presentes a better performance than that of SOM and MIF separately. Several texts from the same author were analyzed. Political analysis texts, sport texts, scientific divulgation texts, scientific texts.

In the first set of experiments, the texts from six authors of daily columns in newspapers were studied. The number of texts for each author varies from 45 to 345, and the length of each text varies from 916 to 1169 words. For each one of the column authors, a small number of texts is selected to define the threshold Θ through SOM. The rest of texts from the author, as well as texts from other column authors, are analyzed in order to decide if the authorship is the same or not. Results are shown as a ROC in fig. 3. This image shows the rate of true positives (TP) as a function of false positives (FP). It is observed that the performance is much better than that of a random clasifier, and better than the SOM used as an anomaly detector scheme, as proposed in [13]. Also, the classification rate is presented for the MIF but with a fixed threshold Θ of $\alpha^*/10$. A perfect authorship detector would climb directly to the point $(0, 1.0)$.

The proposed method is affected by the group of texts that define the threshold. If these texts are very similar with regard to the distribution in the embedded space, then the average distance between centers of mass of those texts will be small. This leads to a small Θ, which may be very restrictive even to texts from the same author. The proper selection of texts that defines an author's stylome is still an open question.

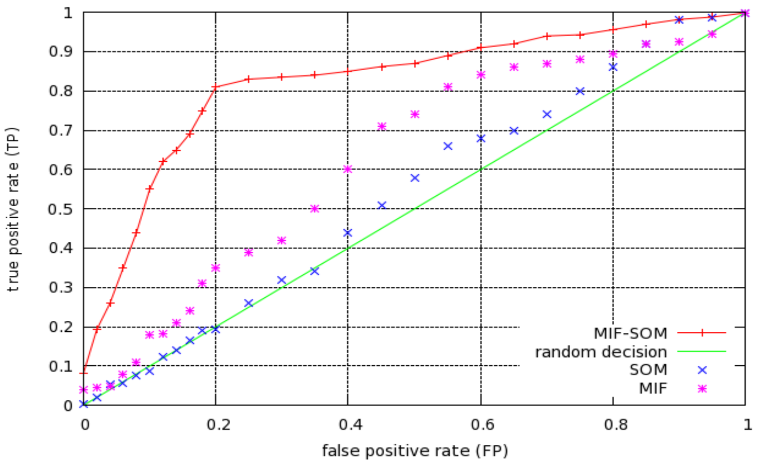


Fig. 3. Average ROC curve for all considered authors. It is shown the behavior of a random decision system and the behavior of our model (MIF-SOM). Also, the performance of SOM and MIF as independent models is shown.

4 Conclusions

Authorship identification is a difficult task, as it is very unlikely that a single method can capture all relevant features an author exhibits in his/her texts, the so called stylome. Thus, we proposed a hybrid model that considers the temporal regularities identified by the mutual information function, tuned by the spatial regularities captured by the self-organzing map.

To validate our model, we assigned to it the task of learning the attributes of a given author through a smalll number of his texts. From those attributes, the meodel should decide if a previously unseen text has the same authorship than the first ones, or not. Our model performed very well fortexts from six different column authors. Although our model is an accurately one, it still has to be tested in more challenging environments. We are evaluating its capabilities to differentiate among several literary styles.

Acknowledgments

This project was partly supported by the Instituto de Ciencia y Tecnologia del Distrito Federal, under project PICCT08-5.

References

1. Malyutov, M.: Authorship attribution of texts: a review. Electronic Notes in Discrete Mathematics 21, 353–357 (2005)
2. Markou, M., Singh, S.: Novelty detection: a review part 1: statistical approaches. Signal Processing 83, 2481–2497 (2003)
3. Markou, M., Singh, S.: Noveltydetection: a review part 2: neural network based approaches. Signal Processing 83, 2499–2521 (2003)
4. Juszczak, P., Tax, D., Pekalska, E., Duin, R.: Minimum spanning tree based one-class classifier. Neurocomputing 72, 1859–1869 (2009)
5. Harmeling, S., Dornhege, G., Tax, D., i Meinecke F, Mueller K.: From outliers to prototypes: ordering data. Neurocomputing 69(13-15), 1608–1618 (2006)
6. Van Halteren, H., Baayen, R., Tweedie, F., Haverkort, M., Neijt, A.: New Machine Learning Methods Demonstrate the Existence of a Human Stylome. Journal of Quantitative Linguistics 12(1), 65–77 (2005), doi:10.1080/09296170500055350
7. Coulthard, M.: Author identification, idiolect, and linguistic uniqueness. Journal of Applied Linguistics 25(4), 431–447 (2004)
8. Clark, J., Hannond, C.: A Classifier System for Author Recognition Using Synonym-Based Features. In: Gelbukh, A., Kuri Morales, Á.F. (eds.) MICAI 2007. LNCS (LNAI), vol. 4827, pp. 839–849. Springer, Heidelberg (2007)
9. Dinu, L., Popescu, M.: Ordinal measures in authorship identification. In: Overview of the 1st International Competition on Plagiarism Detectioni, pp. 1–9 (2009)
10. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer, Heidelberg (2000)
11. Ritter, H.: Self-Organizing Maps on non-euclidean Spaces Kohonen Maps. In: Oja, E., Kaski, S. (eds.) pp. 97–108 (1999)
12. Cottrell, M., Fort, J.C., Pagés, G.: Theoretical aspects of the SOM algorithm. Neurocomputing 21, 119–138 (1998)

13. Barreto, G., Aguayo, L.: Time Series Clustering for Anomaly Detection Using Competitive Neural Networks. In: Príncipe, J.C., Miikkulainen, R. (eds.) WSOM 2009. LNCS, vol. 5629, pp. 28–36. Springer, Heidelberg (2009), doi:10.1007/978-3-642-02397-2
14. Cellucci, C., Albano, A., Rapp, P.: Statistical validation of mutual information calculations: Comparison of alternative numerical algorithms. Physical Review E 71, 066208 (2005)
15. Gross, E., Herzel, H., Buldyrev, S., Stanley, E.: Species independence of mutual information in coding and noncoding DNA. PRE 61(5), 5624–5629 (2000)
16. Bauer, M., Schsuter, S., Sayood, K.: The Average Mutual Information Profile as a Genomic Signature. BMC Bioinformatics 9(48) (2008)

Supervised Machine Learning for Predicting the Meaning of Verb-Noun Combinations in Spanish

Olga Kolesnikova and Alexander Gelbukh

Center for Computing Research, National Polytechnic Institute
Mexico City, 07738, Mexico
kolesolga@gmail.com
www.gelbukh.com

Abstract. The meaning of such verb-noun combinations as *take care*, *undertake work*, *pay attention* can be generalized as DO what is designated by the noun. Likewise, the meaning of *make a decision*, *provide support*, *write a letter* can be generalized as MAKE what is designated by the noun. These generalizations represent the meaning of certain groups of verb-noun combinations. We use supervised machine learning algorithms to predict the meanings DO, MAKE, BEGIN, and CONTINUE of previously unseen verb-noun pairs. We evaluate the performance of the applied algorithms on a training set using 10-fold cross-validation technique. The learnt models have also been evaluated on an independent test set and the predictions have been checked manually to determine the accuracy of the classifiers. The obtained results show that supervised machine learning methods achieve significant accuracy and can be used for semantic annotation of verb-noun combinations.

Keywords: lexical functions, verb-noun combinations, meaning representation by means of hypernyms, supervised machine learning.

1 Introduction

The meaning of individual words can be described by definitions in conventional dictionaries for human usage like Longman Dictionary of Contemporary English or the Merriam-Webster English Dictionary. Often, most frequent words have many senses. For example, Longman Dictionary of Contemporary English [5] gives 47 senses for the verb *take*, 44 for *make*, the number of senses for *have* reaches 49, but *play* looks very poor with only 10 senses! Combinations of verbs with prepositions, called phrasal verbs, like *take after*, *make over*, *have on* etc. are not counted as separate senses otherwise the number of senses would have grown tremendously!

Taking a careful look at definitions of the previously given verbs, one can notice that these verbs have some meanings in common. Note, that we have used word definitions from the Longman Dictionary of Contemporary English mentioned above; therefore in this Section, when referring to the dictionary we mean the Longman Dictionary of Contemporary English.

Coming back to the fact of meaning repetitions in verb definitions, we now give a few examples of verbs which have the meaning DO STH (STH = something) among other senses. First, let us consider the verb *take*. The dictionary gives the following definition of *take* in the sense DO STH: 'a word meaning to do something used with many different nouns to form a phrase that means: "do the actions connected with the nouns": take a walk / take a bath / take a breath / take a vacation.' The second example is the verb *make*. In the dictionary, it also has the sense DO STH followed by the comment: 'used with some nouns to mean that someone performs the action of the noun: make a decision / mistake.' Thirdly, even the verb *have* which is typically used in the sense *possess*, can acquire the meaning DO STH in combination with some nouns. In this meaning, *have* is described as 'a word meaning to do something, used in certain phrases: have a look / walk / sleep / talk / thing / a holiday / bath / shower'.

Lastly, let us consider the verb *play*. One of its meanings given in the dictionary is 'to take part in a game or sport' like golf, chess, etc. Though the exact phrase DO STH or the exact word DO is not encountered in the definition of *play*, we look for the definition of *to take part* in the dictionary and find: *to take part* is 'to do an activity, sport etc. with other people'. Therefore it can be affirmed that *play* also has DO as one of its senses, because in the definition of *play*, we can substitute *to take part* by 'to do an activity, sport etc. with other people'.

We will call the meaning DO STH, or just DO, the generalized meaning of the verbs *take*, *make*, *have*, and *play*, since DO is used in the first, more general, part of the verb definitions. Table 1 gives other examples of the generalized meaning DO. For clarity and illustration, verbs are given in combination with nouns.

Table 1. Verbs with the meaning DO

Verb	Dictionary definition of the sense generalized as DO
give somebody / sth a smile / laugh / shout / push	do something – to smile, laugh, shout etc.: *He gave me a quick smile and a hug.* | *Ooh, the baby just gave a kick*!
conduct a survey / experiment / inquiry etc	to carry out a particular process, especially in order to get information or prove facts: *The company conducted a survey to find out local reaction to the leisure center.*
carry sth ***out***	to do something that needs to be organized and planned: *They are carrying out urgent repairs.* | *A survey is now being carried out nationwide.* | *It won't be an easy plan to carry out.*
ask (a question)	to say or write something in order to get an answer, a solution, or information: *That kid's always asking awkward questions.*
teach	to give lessons in a school, college, or university: *The guy's been teaching in France for 3 years now.*

Likewise, other generalized meaning can be determined. In this work, we are interested in generalized meanings DO, MAKE, BEGIN and CONTINUE. We do not give formal definitions for MAKE, BEGIN and CONTINUE but illustrate them with examples in Tables 2, 3, and 4, respectively. The examples are chosen to represent the given meanings in an exact and comprehensive way.

It is a generally accepted fact that the meaning of an individual word depends on its context, i.e. the words it is used with in corpora. This fact is also true in the case of generalized meanings that we have selected. Verbs acquire these meanings when collocate with nouns belonging to a particular semantic group, for example, a group denoting actions. If verb-noun combinations are annotated with the meanings DO, MAKE, BEGIN or CONTINUE, this annotation disambiguate both the verb and the noun. Word sense disambiguation is one of the most important and challenging tasks of natural language processing, and therefore semantic annotation of verb-noun combinations is a task of significant relevance.

It should be noted here that the concept of generalized meaning we propose here is close to the notion of lexical functions developed by the Meaning-Text Theory. Lexical function is a mapping from one word (called **keyword**, for example, *decision*) to another it collocates with in corpora (called **lexical function value**). This mapping is further characterized by the meaning of semantically homogenuous groups of values and by typical syntactic patterns in which lexical function values are used with their respective keywords in texts. For the keyword *decision*, the lexical function Oper$_1$, meaning 'do, perform, carry out', gives the value *make*. That is, to express the meaning 'do, or perform, a decision', one says in English *make a decision*. The formalism of lexical functions is intended to represent fixed word combinations, or collocations like *make a decision*, *give a lecture*, *lend support*, etc. For more information on lexical functions consult [6, 7]. We do not apply the formalism of lexical functions as it is. Our purpose is to predict semantic contents of verb-noun combinations, and the meanings we have chosen, are not exactly the meanings of lexical functions though have some resemblance to them. Another difference is that lexical functions describe collocations, but generalized meanings are present in collocations as well as in free word combinations. Section 3 gives details concerning state-of-the-art research on lexical functions.

The rest of the paper is organized as follows. Section 2 formulates our task. Section 3 gives a summary of related work. Section 4 explains what data was used in the experiments. Section 5 describes methodology. We present results and discuss them in Section 6. Section 7 outlines conclusions and future work.

Table 2. Verbs with the meaning MAKE

Verb	Dictionary definition of the sense generalized as MAKE
create	to make something exist that did not exist before: *Her behaviour was creating a lot of problems.*
cause	to make something happen: *Heavy traffic is causing long delays on the freeway.*
build	to make something, especially a building or something large: *Are they going to build on this land?*
write	to produce a new book, poem, song etc.
produce	to make things to be sold: *Gas can be produced from coal.*

Table 3. Verbs with the meaning BEGIN

Verb	Dictionary definition of the sense generalized as BEGIN
start	to begin doing something: *start learning German / work*
enter	to start working in a particular profession or organization: *Andrea is studying law as a preparation for entering politics.*
introduce	be the start of; if an event introduces a particular period or change, it is the beginning of it: *The death of Pericles in 429 BC introduced a darker period in Athenian history.*
launch	to start something, especially an official, public, or military activity that has been carefully planned: *launch a campaign / appeal / inquiry*
become	to begin to be something: *He became King at the age of 17.*

Table 4. Verbs with the meaning CONTINUE

Verb	Dictionary definition of the sense generalized as CONTINUE
keep	to continue to have something and not lose it or get rid of it: *No, we're going to keep the house in Vermont and rent it out.*
maintain	to make something continue in the same way or at the same high standard as before: *Britain wants to maintain its position as a world power.*
pursue	to continue doing an activity or trying to achieve something over a long period of time: *Kristin pursued her acting career with great determination.*
sustain	to mak something continue to exist over a period of time: *The teacher tried hard to sustain the children's interest.*
run	to continue to be officially able to be used for a particular period of time: *The contract runs for a year.*

2 Task

The task of our work is to examine performance of supervised learning methods for prediction of the meanings DO, MAKE, BEGIN, and CONTINUE in Spanish verb-noun combinations. We train classifiers on a manually compiled corpus of verb-noun pairs annotated with the above given meanings. After building classification models on the training data, the models are tested for prediction of the meanings on unseen data. The data used for testing the models are of two types. The first type of the testing data is a part of the training set which is divided into the training section and the test section applying the 10-fold cross-validation technique. The second type of testing data is an independent test set build on a corpus other than the corpus used to construct the training set. The details concerning data can be found in Section 4.

3 Related Work

The meaning of word combinations is often represented as a semantic relation between individual words that constitute word combinations. We will give a short review of two lines of research devoted to semantic relations.

The first line is work on automatic detection of lexical functions mentioned in the Introduction. Lexical functions are semantic relations which hold between constituents of fixed word combinations, or collocations. Collocations may have different syntactic structures, and the verb-noun pattern is one of these structures. L.Wanner [16, 17] made experiments to classify Spanish verb-noun pairs according to nine lexical functions with the meaning 'perform, experience, carry out something, 'cause the existence of something, 'begin to perform something, 'continue to perform something', etc. Verb-noun pairs were divided in two groups. In the first group, nouns belonged to the semantic field of emotions; in the second groups nouns were field-independent. For classification, the following supervised learning algorithms were applied: Nearest Neighbor technique, Naïve Bayesian network, Tree-Augmented Network Classification technique and a decision tree classification technique based on the ID3-algorithm. As a source of information for building the training and test sets, hyperonymy hierarchy of the Spanish part of EuroWordNet [15, 12] was used. The average f-measure of about 70% was achieved in these experiments. The best results for field-independent nouns were shown by ID3 algorithm (f-measure of 0.76) for the lexical function with the meaning ‘cause (by the noun functioning in utterances as the verb’s direct object) something to be experienced / carried out / performed’ and by the Nearest Neighbor technique (f-measure of 0.74) for the lexical function with the meaning ‘perform / experience / carry out something’.

The second line of research on semantic relations in word combinations deals with automatic assignment of semantic relations to English noun-modifier pairs in [8, 9]. Though in our work, verb-noun combinations are treated, we believe that the principles of choosing data representation and machine learning techniques for detection of semantic relations between a noun and a modifier can also be are used to detect semantic relations in verb-noun pairs. The underlying idea is the same: learning the meaning of word combinations. In [8, 9], the researchers examined the following relations: causal, temporal, spatial, conjunctive, participant, and quality. They used two different data representations: the first is based on WordNet relations, the second, on contextual information extracted from corpora. They applied memory-based learning, decision tree induction and Support Vector Machine. The highest f-score of 0.847 was achieved by C5.0 decision tree to detect temporal relation based on WordNet representation.

4 Data

Verb-noun pairs were extracted automatically from the Spanish Web Corpus [11] by the Sketch Engine [4] and ranked by frequency. Thus we obtained a list of 83, 982 pairs. From this list, we have taken the first one thousand pairs and processed them manually as follows.

First, we removed all fallacious combinations extracted from the Spanish Web Corpus automatically due to parsing errors. Erroneous pairs included, for instance, past participles or infinitives instead of nouns, or contained symbols like --, « , © instead of words. The total number of erroneous pairs was 61, so after their removal the list contained 939 pairs.

Secondly, we disambiguated each verb and noun, annotating them with word senses of the Spanish WordNet [15, 12]. For some verb-noun pairs, relevant senses were not found in the above mentioned dictionary, and the number of such pairs was 39. For example, in the combination *dar cuenta*, ‘give account’, the noun *cuenta* means *razón*, *satisfacción de algo*, ‘reason, or satisfaction of something’. This sense of *cuenta* is taken from *Diccionario de la Lengua Española*, ‘Dictionary of the Spanish Language’ [2]. Unfortunately, this sense is absent in the Spanish WordNet so the expression *dar cuenta* was left without sense annotation. All combinations that could be not annotated with senses of the Spanish WordNet were removed from the list.

After the first two steps, 900 verb-noun pairs were left in the list. We have looked through the list and annotated all relevant combinations with the meanings DO, MAKE, BEGIN, and CONTINUE. We found 280 pairs with the meaning DO, 112 pairs with the meaning MAKE, BEGIN was encountered in 25 pairs, and CONTINUE was observed to be the most rare meaning with only 16 verb-noun pairs. Thus the total number of verb-noun pairs annotated with four meanings was 433, and 467 pairs had meanings other than DO, MAKE, BEGIN, CONTINUE. All 900 pairs were included in the training sets. Table 5 demonstrates examples of the data. The examples are given as they are encountered in the list built automatically, so the nouns are used without articles or quantifiers.

We build four training sets, one for each of the four meanings. All training sets included the same 900 examples which were marked differently depending on the meaning chosen for a given set. For example, the training set for DO included 280 positive instances marked as the class "yes" and the rest of the examples (620 instances) were marked as the class "no", i.e. these were instances of the meanings MAKE, BEGIN, and CONTINUE, as well as the verb-noun pairs with meaning other that DO, MAKE, BEGIN, CONTINUE.

Table 5. Examples of verb-noun pairs

Meaning	**Examples**	
	Spanish	**English lit. translation**
DO	*hacer justicia* *realizar actividad* *dar beso*	*do justice* *realize activity* *give kiss*
MAKE	*hacer ruido* *establecer criterio* *encontrar solución*	*make noise* *establish criterion* *find solution*
BEGIN	*iniciar proceso* *tomar iniciativa* *adoptar actitud*	*initialize process* *take initiative* *adopt attitude*
CONTINUE	*mantener control* *llevar vida* *seguir curso*	*maintain control* *lead life* *follow course*

Lastly, for each verb and noun in the training sets, we extracted all hyperonyms from the Spanish WordNet. We represented each verb-noun pair as a set of all hyperonyms of the noun and all hyperonyms of the verb. Both constituents of verb-noun pairs were considered as zero-level hyperonyms, they were also included in the set of hyperonyms.

To build an independent test set, we extracted 5181 verb-noun pairs from the Spanish Treebank Cast3LB [1], a corpus other than the corpus used to construct the training sets. To evaluate the performance of classifiers, we used the test set in the following ratios: 100%, 75%, 50%, and 25%.

We did not disambiguate verb-noun pairs for the test sets manually. Instead, for each verb-noun, we built all possible verb-noun combinations of all senses in the Spanish WordNet. As an example, let us consider the pair *representar papel*, 'represent role'. The verb *representar* has 12 senses in the Spanish WordNet, and the noun *papel*, 5. This gives totally 60 combinations of *representar* and *papel* (12 multiplied by 5). Remember, that the test data included totally 5,181 verb-noun pairs which resulted in 96,079 instances in the test set.

The training and test sets were formatted according to Attribute-Relation File Format (ARFF) [14] to be accessible by machine learning methods described in Section 5. Every hyperonym was presented as an attribute with two possible values, "1" if a corresponding hyperonym is encountered in a particular verb-noun pair, and "0" if it is not. Thus each verb-noun pair was represented as a vector of zeros and ones. The last attribute was a categorical feature with two possible values, “yes” if a corresponding verb-noun pairs has the meaning that is to be learnt by classifiers, and “no” if it is not.

5 Methodology

Our approach is based on supervised machine learning algorithms as implemented in the WEKA version 3-6-2 toolset [13, 3, 18]. We performed two groups of experiments. In the first group of experiments, we evaluated the prediction of the meanings DO, MAKE, BEGIN, and CONTINUE on the training sets using 10-fold cross-validation technique. In the second group of experiments, the same meanings were predicted for the instances of an independent test set. Table 6 lists all classifiers we experimented with.

Table 6. Classifiers

Classifier	Classifier	Classifier
AODE	ClassificationViaClustering	VFI
AODEsr	ClassificationViaRegression	ConjunctiveRule
BayesianLogisticRegression	CVParameterSelection	DecisionTable
BayesNet	Dagging	JRip
HNB	Decorate	NNge
NaiveBayes	END	OneR
NaiveBayesSimple	EnsembleSelection	PART
NaiveBayesUpdateable	FilteredClassifier	Prism
WAODE	Grading	Ridor
LibSVM	LogitBoost	ZeroR
Logistic	MultiBoostAB	ADTree
RBFNetwork	MultiClassClassifier	BFTree
SimpleLogistic	MultiScheme	DecisionStump
SMO	OrdinalClassClassifier	FT
VotedPerceptron	RacedIncrementalLogitBoost	Id3
Winnow	RandomCommittee	J48
IB1	RandomSubSpace	J48graft
IBk	RotationForest	LADTree
KStar	Stacking	RandomForest
LWL	StackingC	RandomTree
AdaBoostM1	ThresholdSelector	REPTree
AttributeSelectedClassifier	Vote	SimpleCart
Bagging	HyperPipes	

6 Experimental Results

6.1 Experiments on the Training Sets

The purpose of our experiments was to evaluate performance of 68 classifiers on the training sets using 10-fold cross validation technique. The best five results for predicting the "yes" class are presented in Tables 7, 8 (remember, "yes" and "no" classes are explained in Section 4 alongside with other details about data). P stands for precision, R for recall and F for *f*-measure. Together with the best results, Table 7, 8 show performance of the four classifiers most frequently used in natural language processing, i.e. support vector machine (implemented in WEKA as SMO), C4.5 decision tree learner (J48 in WEKA), Naive Bayes algorithm, and nearest-neighbor instance-based learner (IB1 in WEKA). These 4 classifiers are left in the tables ranked by *f*-measure so it can be seen what algorithm is better for detecting each meaning.

It is a common practice that in classification experiments, the performance of rules.ZeroR classifier is considered as the baseline. ZeroR is a trivial algorithm that always predicts the majority class. But in our training sets the majority class is always the class of negative examples. Remember, that the overall number of positive and negative instances in the training sets is 900, though the largest number of positive instances is 280 for the meaning DO which still is much less then the number of negative instances (620 in the case of DO). Therefore, ZeroR does not classify any test instances as positives, which always gives recall of 0 and undefined precision. For this reason, ZeroR should not be considered as the baseline.

Table 7. Performance of WEKA Classifiers on DO and MAKE training sets

DO				MAKE			
Classifier	P	R	F	Classifier	P	R	F
PART	0.898	0.857	**0.877**	JRip	0.726	0.706	0.716
SimpleCart	0.901	0.853	0.876	SimpleCart	0.728	0.688	0.708
BLR	0.875	0.872	0.874	LADTree	0.706	0.706	0.706
Bagging	0.884	0.857	0.870	REPTree	0.721	0.688	0.704
BFTree	0.903	0.838	0.869	BFTree	0.730	0.670	0.699
...	...	...	...	...	...	...	...
SMO	0.856	0.872	0.864	SMO	0.689	0.651	0.670
J48	0.876	0.850	0.863	J48	0.747	0.541	0.628
NaiveBayes	0.762	0.711	0.735	IB1	0.532	0.376	0.441
IB1	0.566	0.759	0.648	NaiveBayes	0.535	0.211	0.303

The best classifiers for prediction of the meaning DO is PART, for the meaning MAKE, JRip, for BEGIN, Prism, and for CONTINUE, Ridor. All four classifiers are rule based classification algorithms. Inductive rule learning use separate-and-conquer strategy. It means, that a rule that works for many instances in the class is identified

Table 8. Performance of WEKA Classifiers on BEGIN and CONTINUE training sets

BEGIN				CONTINUE			
Classifier	P	R	F	Classifier	P	R	F
Prism	0.778	0.737	0.757	Ridor	0.813	0.813	0.813
FT	0.762	0.667	0.711	REPTree	0.857	0.750	0.800
SMO	0.824	0.583	0.683	LWL	0.857	0.750	0.800
VFI	0.750	0.625	0.682	EnsembleSelection	0.857	0.750	0.800
NNge	0.750	0.625	0.682	RandomSubSpace	0.857	0.750	0.800
...	...	...	...	...	...	...	...
JRip	0.667	0.500	0.571	J48	0.750	0.750	0.750
IB1	0.818	0.375	0.514	SMO	0.786	0.688	0.733
NaiveBayes	0.000	0.000	0.000	IB1	0.500	0.313	0.385
Prism	0.778	0.737	0.757	NaiveBayes	0.000	0.000	0.000

first, then the instances covered by this rule are excluded from the training set and the learning continues on the rest of the instances. These learners are efficient on large, noisy datasets. Our training sets included 900 instances represented as vectors of the size 1109 attributes, and rule induction algorithms performed very well.

The best state-of-the-art result for predicting a lexical function with the meaning ‘cause’ is f-measure of 0.76 given by ID3 algorithm [17]. In our experiments, the best f-measure of 0.877 was shown by PART for the meaning MAKE. However, such a comparison is not fair, since our task was to predict the meanings DO, MAKE, BEGIN, CONTINUE but not lexical functions as explained in the Introduction.

6.2 Experiments on the Test Sets

Some of the best classifiers displayed in Tables 7, 8 were evaluated on an independent test set built as described in Section 4. Tables 9, 10 present the results for these classifiers. We listed the values of precision, recall and *f*-measure for each classifier in this way: <precision>|<recall>|<f-measure>; BLR in the column **Classifier** stands for BayesianLogisticRegression. As we explain below, the test sets had such a big size that some classifiers failed to make predictions within a reasonable time period. For such classifiers, we put N/A instead of metrics as for the other classifiers.

It was mentioned in Section 4, that since we did not disambiguate verb-noun pairs in the test sets, for each pair we build the number of instances equal to the number of senses for the verb multiplied by the number of senses for the noun. This has given us 96079 instances and 10544 attributes in 100% test set, 73021 instances and 9495 attributes in 75% test set, 48904 instances and 8032 attributes in 50% test set, and 22254 instances and 5857 attributes in 25% test set. SimpleCart, FT, LWL had difficulties in predicting the value of the class variable on test sets of sizes more than 25%. Among these three classifiers, SimpleCart was better because this algorithm was effective enough to process a 75% and 50% set. SimpleCart and FT are decision tree algorithms, and LWL is a nearest-neighbor instance-based learner. Note, that almost all the best classifiers that could process a full-size test set, belong to the class rules. BayesianLogisticRegression also performs well and the only algorithm of the class trees that did not experience time problems was LADTree.

Table 9. Performance of WEKA classifiers on the test set

Meaning	**Classifier**	**Test set size**	
		100%	**75%**
DO	PART	0.261|0.864|0.400	0.304|0.864|0.382
	SimpleCart	N/A	N/A
	BLR	0.178|0.830|0.293	0.212|0.818|0.337
MAKE	JRip	0.231|0.662|0.342	0.189|0.662|0.294
	SimpleCart	N/A	0.168|0.662|0.268
	LADTree	0.285|0.676|0.401	0.168|0.676|0.269
BEGIN	FT	N/A	N/A
	SMO	0.331|0.793|0.467	0.567|0.793|0.661
	NNge	0.302|0.724|0.426	0.567|0.724|0.636
CONTINUE	Ridor	**0.799**|0.480|0.600	**0.993**|0.480|0.647
	REPTree	0.581|0.480|0.526	**0.820**|0.480|0.606
	LWL	N/A	N/A

Table 10. Performance of WEKA classifiers on the test set

Meaning	**Classifier**	**Test set size**	
		50%	**25%**
DO	PART	0.245|0.864|0.382	0.162|0.852|0.272
	SimpleCart	0.405|0.864|0.551	0.281|0.852|0.423
	BLR	0.205|0.818|0.328	0.145|0.807|0.246
MAKE	JRip	0.189|0.662|0.294	0.174|0.662|0.276
	SimpleCart	0.203|0.662|0.311	0.177|0.662|0.279
	LADTree	0.144|0.676|0.237	0.177|0.676|0.281
BEGIN	FT	N/A	0.409|0.724|0.523
	SMO	0.464|0.793|0.585	0.404|0.793|0.535
	NNge	0.603|0.724|**0.658**	0.451|0.724|0.556
CONTINUE	Ridor	**0.958**|0.480|0.640	**1.000**|0.480|**0.649**
	REPTree	0.694|0.480|0.567	0.667|0.480|0.558
	LWL	N/A	**1.000**|0.480|**0.649**

As it is seen from Tables 9, 10, the best precision was shown by Ridor. This method (Ridor = RIpple-DOwn Rule learner) have been developed for knowledge acquisition where it is hard to add a new rule and be sure that it would not cause the inconsistency of the rules generated before. Ridor algorithm is different from covering algorithms for constructing the rule set; instead it generates exceptions for the existing rules that work within the confines of these rules thus not affecting other rules. Then it iterates on the exceptions for the best solution. This scheme allowed the classifier to reach 100% precision. Unfortunately, it can not boast the best recall which is only 0.649 for the meaning CONTINUE on a 25% test set. Still, it is the second best recall in our experiments on test sets. The top recall is 0.658 shown by NNge for the meaning BEGIN on a 50% test set.

Another classifier that gives the best precision of 100% is LWL when performing predictions for the meaning CONTINUE on a 25% test set. But, like Ridor, it shows the same low recall of 0.658. However, a high precision of Ridor and LWL makes them appropriate for fulfilling the tasks where precision is of special importance, for example, for automatic construction of dictionaries.

7 Conclusions and Future Work

We have shown that it is feasible to apply machine learning methods as implemented in the WEKA toolkit for predicting the meaning of unseen Spanish verb-noun collocations. In particular, we trained classifiers to assign the meanings DO, MAKE, BEGIN and CONTINUE to a previously unseen verb-noun pair.

Verb-noun pairs were represented as sets of hyperonyms for both the verb and the noun. As our experiments have shown, hyperonyms function sufficiently well as features distinguishing between the meanings we chosen to be predicted by classifiers.

The best f-measure achieved in our experiments is 0.877 using the training set and 10-fold cross-validation technique. This is significantly higher than the previously reported result of 0.740 for f-measure, though the comparison is not fair because we looked for the meaning which is similar to the meaning predicted in [17], but not the same one. The highest f-measure achieved in the experiments on an independent test set was only 0.658. This could be explained by the fact that the best ratio between the training set and the test set has not yet been found by us. More experiments on test sets of various sizes are needed.

In the future, we plan to test other classification methods that were not examined in our experiments as well as to work with data extracted from a raw corpus and lemmatized [10]. We also plan to study the effect of other features, such as WordNet glosses and to make experiments with word space models representing various similarity measures between word combinations. We will experiment with different ratios between the training set and the test set.

Acknowledgements. We are grateful to Adam Kilgarriff and Vojtěch Kovář for providing us a list of most frequent verb-noun pairs from the Spanish Web Corpus of the Sketch Engine, www.sketchengine.co.uk.

The work was done under partial support of Mexican Government: SNI, COFAA-IPN, PIFI-IPN, CONACYT grant 50206-H, and SIP-IPN grant 20100773.

References

1. Civit, M., Martí, M.A.: Building Cast3LB: A Spanish Treebank. Research on Language and Computation 2(4), 549–574 (2004)
2. Diccionario de la Lengua Española. Real Academia Española, Madrid (2001)
3. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11(1) (2009)
4. Kilgarriff, A., Rychly, P., Smrz, P., Tugwell, D.: The Sketch Engine. In: Proceedings of EURALEX 2004, pp. 105–116 (2004)
5. Longman Dictionary of Contemporary English, 3rd edn. Longman Group Ltd., Essex (1995)
6. Mel'čuk, I.A.: A Theory of the Meaning-Text Type Linguistic Models. Nauka Publishers, Moscow (1974) (in Russian)

7. Mel'čuk, I.A.: Lexical Functions: A Tool for the Description of Lexical Relations in a Lexicon. In: Wanner, L. (ed.) Lexical Functions in Lexicography and Natural Language Processing, pp. 37–102. Benjamins Academic Publishers, Amsterdam (1996)
8. Nastase, V., Szpakowicz, S.: Exploring noun-modifier semantic relations. In: 5th International Workshop on Computational Semantics (IWCS-5), Tilburg, Netherlands, pp. 285–301 (2003)
9. Nastase, V., Sayyad-Shiarabad, J., Sokolova, M., Szpakowicz, S.: Learning noun-modifier semantic relations with corpus-based and wordnet-based features. In: Proceedings of the Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, AAAI Press, Menlo Park (2006)
10. Sidorov, G.: Lemmatization in automatized system for compilation of personal style dictionaries of literature writers. In: Word of Dostoyevsky, pp. 266–300. Russian Academy of Sciences, Moscow (1996)
11. Spanish Web Corpus, `http://trac.sketchengine.co.uk/wiki/Corpora/SpanishWebCorpus/` (last viewed June 02, 2010)
12. Spanish WordNet, `http://www.lsi.upc.edu/~nlp/web/index.php?Itemid=57&id=31&option=com_content&task=view` (last viewed June 02, 2010)
13. The University of Waikato Computer Science Department Machine Learning Group, WEKA download, `http://www.cs.waikato.ac.nz/~ml/weka/index_downloading.html` (last viewed June 02, 2010)
14. The University of Waikato Computer Science Department Machine Learning Group, Attribute-Relation File Form, `http://www.cs.waikato.ac.nz/~ml/weka/arff.html` (last viewed June 02, 2010)
15. Vossen, P. (ed.): EuroWordNet: A Multilingual Database with Lexical Semantic Networks. Kluwer Academic Publishers, Dordrecht (1998)
16. Wanner, L.: Towards automatic fine-grained classification of verb-noun collocations. Natural Language Engineering 10(2), 95–143 (2004)
17. Wanner, L., Bohnet, B., Giereth, M.: What is beyond Collocations? Insights from Machine Learning Experiments. In: EURALEX (2006)
18. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)

On the Structure of Elimination Trees for Bayesian Network Inference

Kevin Grant and Keilan Scholten

Department of Math and Computer Science
University of Lethbridge
Lethbridge, Alberta, Canada
{kevin.grant2,keilan.scholten}@uleth.ca

Abstract. We present an optimization to elimination tree inference in Bayesian networks through the use of *unlabeled* nodes, or nodes that are not labeled with a variable from the Bayesian network. Through the use of these unlabeled nodes, we are able to restructure these trees, and reduce the amount of computation performed during the inference process. Empirical tests show that the algorithm can reduce multiplications by up to 70%, and overall runtime by up to 50%.

Keywords: Bayesian networks, inference, offline computation.

1 Introduction

Recursive decomposition refers to a class of algorithms for computing probabilities from Bayesian networks (henceforth known as *inference*). In recursive decomposition [10,3,6], variables in the network are conditioned, which decomposes the network into smaller and smaller problems. These smaller problems are then solved, and their results combined to compute a global solution.

Recursive decomposition algorithms typically have an associated tree structure, either implicit or explicit, that determines the order in which computations take place. These trees differ from technique to technique, and each tree type has a specific set of properties. For example, a dtree [3] is a full binary tree - all non-leaf nodes have exactly two children. By contrast, elimination trees [6] have no restriction on the number of children each node can have, although each internal node must be labeled by exactly one variable. These constraints on structure affect how probabilities are computed, and each has its specific advantages.

In this paper, we consider a more general structure for elimination trees, in order to optimize their associated inference algorithm. Specifically, we use *unlabeled nodes* (a technique borrowed from dtrees), or nodes that are not labeled by a variable from the Bayesian network. These unlabeled nodes allow the children of a labeled internal node to be restructured from their current flat model into a more general model, without affecting the current variable elimination ordering. Through this restructuring, we are able to reduce both the number of recursive calls and arithmetic operations that occur during the inference process, lowering overall runtime. Note that this optimization occurs with no effect to the actual inference algorithm (hence maintaining its simplicity), and with only a small effect on memory usage. While this paper focuses specifically on elimination trees, these techniques can be applied to other recursive models (e.g. the *compose* operator for dtree construction [3]).

2 Background and Previous Work

We denote random variables with capital letters (e.g. X, Y, Z), and sets of variables with boldfaced capital letters $\mathbf{V} = \{V_1, ..., V_n\}$. Each random variable V has a discrete domain of finite size $|V|$ containing values $\mathcal{D}(V) = \{0, ..., |V| - 1\}$. An instantiation of a variable is denoted $V{=}v$, or v for short, where $v \in \mathcal{D}(V)$. A *context*, or instantiation of a set of variables, is denoted $\mathbf{X}{=}\mathbf{x}$ or simply $\mathbf{x}$.

Given a set of random variables $\mathbf{V} = \{V_1, ..., V_n\}$ with domain function $\mathcal{D}$, a Bayesian network is a tuple $\langle \mathbf{V}, \mathbf{\Phi} \rangle$. $\mathbf{\Phi} = \{\phi_{V_1}, ..., \phi_{V_n}\}$ is a set of probability distributions with a one-to-one correspondence with the elements of $\mathbf{V}$. A Bayesian network has an associated *directed acyclic graph* (DAG), and each $\phi_{V_i} \in \mathbf{\Phi}$ is the conditional probability of V_i given its parents in the DAG (called *conditional probability tables* or CPTs). That is, if π_{V_i} represents the parents of V_i, then $\phi_{V_i} = P(V_i|\pi_{V_i})$. The *definition* of a discrete variable function is the set of variables over which it is defined. The *family* of a variable V in a Bayesian network is defined as V plus the parents of V. Figure 1 shows the DAG of a Bayesian network, taken from [11].

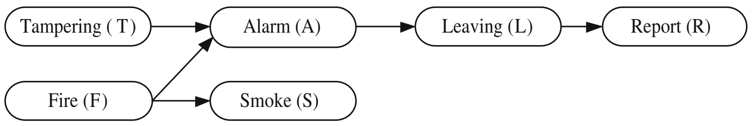


Fig. 1. An example Bayesian network

Inference in Bayesian networks typically refers to the computation of probabilities from the Bayesian network, given a set of observations. As a simple example, given the Bayesian network in Figure 1, we may be interested in computing $P(F = 1|R = 1)$, i.e. the probability that a fire is occurring given that people have been reported leaving the building. Inference in Bayesian networks is NP-hard [2], however, many algorithms have been proposed that provide reasonable runtimes under many circumstances. Three of the more popular classes of algorithms include Junction Tree Processing [8], Variable Elimination [12], and Recursive Decompositions [3,6]. Each class of algorithms has its own specific advantages, depending on the application. Space precludes a more detailed discussion of these inference techniques; the interested reader is referred to the cited references for more information.

In this work, we focus on recursive decompositions. This class of algorithms compiles the Bayesian network into a tree structure, and computes probabilities

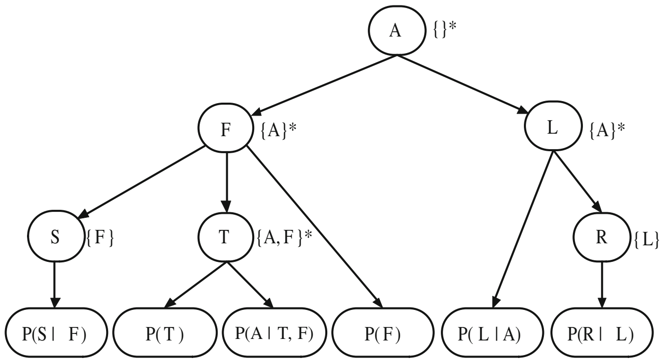


Fig. 2. The network from Figure 1, arranged in an etree

using a recursive algorithm. One of the primary advantages of recursive decomposition algorithms is that they permit a straightforward time-space tradeoff, making them usable in memory-constrained environments. This paper focus on on elimination trees; however, much of the discussion and proposed techniques can be applied to other recursive inference techniques as well.

An *elimination tree* (etree) [6] is a tree whose leaves and internal nodes correspond to the CPTs and variables of a Bayesian network, respectively. The tree is structured such that all CPTs containing variable V_i in their definition are contained in the subtree of the node labelled with V_i. Figure 2 shows one of the possible etrees for the Bayesian network of Figure 1.

The function $\mathcal{P}$ (Figure 3) recursively computes the probability of a given context over a subset of variables in an etree, using a top-down approach. We use the following notation: if T is a leaf node, then ϕ_T represents the CPT at T. If T is an internal node V_T represents the variable labeling T, and ch_T represents its children. Finally, if an etree $\mathcal{T}$ is used in the context of a tree node, then we are referring to the root node of $\mathcal{T}$.

The time complexity of $\mathcal{P}$ over an etree T is exponential on T's height, while the space complexity is exponential only on the largest family in the Bayesian network [6]. However, this complexity can be improved through a technique called *caching* [3]. Let N_V denote the internal node in an etree that is labeled with variable V. Consider the tree from Figure 2, and consider calling node N_S when $A = 0$ and $F = 0$:[1]

$$\mathcal{P}(N_S, \{A = 0 \wedge F = 0\}) = \sum_{s \in \mathcal{D}(S)} P(S = s | F = 0) \tag{1}$$

Notice that this equation does not depend on the value of A. Hence, when $A = 1$ and $F = 0$, the value returned from node N_S will be exactly the same. By caching this value at node N_S, it needs only be calculated when $A = 0$, and retrieved when $A = 1$.

[1] In this particular example, $\mathcal{P}(N_S, \{A = 0 \wedge F = 0\})$ will always return 1. However, this will not be the case if S is observed.

```
$\mathcal{P}(T, \mathbf{c})$
1.  **if** $T$ is a leaf node
2.    **return** $\phi_T(\mathbf{c})$
3.  **elseif** $V_T$ is instantiated in $\mathbf{c}$
4.    $Total \leftarrow 1$
5.    **for each** $T' \in ch_T$
6.      $Total \leftarrow Total * \mathcal{P}(T', \mathbf{c})$
7.  **else**
8.    $Total \leftarrow 0$
9.    **for each** $v_T \in \mathcal{D}(V_T)$
10.     $Product \leftarrow 1$
11.     **for each** $T' \in ch_T$
12.       $Product \leftarrow Product * \mathcal{P}(T', \mathbf{c} \wedge \{v_T\})$
13.     $Total \leftarrow Total + Product$
14. **return** $Total$
```

Fig. 3. Code for processing an etree given a context

Define the *a-cutset* of node N to be the set of variables labeling the nodes in N's ancestry; the *cache-domain* of N (denoted $CD(N)$, called a *context* in [3]) is the intersection of N's a-cutset and the domains of the CPTs in N's subtree. The cache-domains of each node in Figure 2 are shown in curly braces to the right of each node. The return value from N depends only on the assignment to its cache-domain, and not its a-cutset. This is demonstrated in the previous example: the cache-domain of node N_S is $\{F\}$, since the cache values of this node do not depend on A.

In order to avoid recomputing values, each internal node maintains a cache of its computed values. The cache of N can be implemented as a function over N's cache-domain, and the algorithm for calculating probabilities from an etree (Figure 3) can be very simply modified to perform caching. When a node is visited, we check to see if the corresponding value is cached. If it is, the cached value is returned; if not, the value is calculated, cached, and returned. Caching in recursive decompositions reduces the complexity of inference from being exponential on height, to being exponential on the size of the cache-domains at each node. The sizes of the cache-domains are determined by the *width* of the variable elimination ordering associated with the tree. For more details regarding caching, please see [3]. For the remainder of the document, we will assume that our etree nodes maintain caches.

Dead Caches. In recursive decompositions, dead caches are caches whose values are only computed and never queried [1]. Consider the etree in Figure 2; in particular, consider node N_T. The cache-domain at this node is $\{A, F\}$. When caching is employed, algorithm $\mathcal{P}$ visits a node once for each assignment of its parent's cache-domain and labeling variable. Hence, node N_T is visited only once for each assignment of its own cache-domain. Therefore, the cache values are never actually used, only set. Dead caches can be removed from recursive decompositions with no runtime consequence. A cache is dead if its cache-domain is equivalent to, or a superset of, its parent cache. Because of this simple test, determining dead caches can be done offline. In our etree illustrations, dead caches are marked with an asterisk.

Measuring Inference Cost. Our primary goal is to optimize the etree structure with regards to runtime. We focus on reducing some of the dominant operations that occur during inference, the idea being that this will translate to an overall reduction in runtime.

Our first peformance metric is the number of recursive calls made by $\mathcal{P}$ while performing inference. This metric has been considered in previous publications [1], and we have found this number to have a reasonable correlation with runtime. We will also use number of floating-point multiplications when determining performance, as a secondary measure to break ties when necessary.

One attractive property of these performance metrics is that they can be computed quickly using simple formulae, without having to actually perform inference. The number of recursive calls made at etree node N is:

$$RC(N) = |CD(N)| * |V_N| * |ch_N| \tag{2}$$

where the three multiplicands are the number of possible contexts of N's cache domain, the number of states that N's labeling variable can take, and the number of children that N has, respectively. Note that if N is a leaf, it makes no recursive calls, so $RC(N) = 0$ in this case. The number of floating-point multiplications performed at node N during inference can be computed in a similar manner:

$$FM(N) = |CD(N)| * |V_N| * (|ch_N| - 1) \tag{3}$$

While these metrics will be used by our algorithm when searching for an optimal structure, we will also measure runtime when we evaluate the resulting etrees.

2.1 Related Work

As mentioned in the introduction, there are other recursive decomposition methods for Bayesian network inference. Each technique typically has its own tree structure. One of the more well-known recursive decomposition techniques is *Recursive Conditioning* [3], which uses a tree structure called a dtree. Like an etree, each leaf of a dtree corresponds to a CPT from the Bayesian network, and the internal nodes of the dtree are labeled with variables from the network. Unlike etrees, however, the internal nodes of a dtree must be binary, and there is no restriction on the number of variables that label each node. The algorithm for computing over dtrees and etrees is similar, and it has been shown that for a dtree, an etree of the same time complexity can be constructed [7]. One of the advantages of etrees vs. dtrees is in their inference algorithm; the restriction of a single variable to each internal node permits a very simple algorithm, whose low-level properties make it portable, and accessible to non-experts.

There have been several techniques proposed for dtrees that attempt to optimize their runtime. In [3], the authors present a method for balancing existing dtrees, while in [4], the authors present a method for directly constructing dtrees of low height. While optimizing tree structure is the primary focus of this paper, the primary difference between these techniques and those presented here is that their optimizations attempt to minimize the height of the dtree, which optimizes runtime for a non-caching model, whereas our optimization techniques ignore tree height, and are designed primarily to optimize caching models.

The restructuring that we will describe is closely related to the *optimal factoring problem* (OFP) [9], where inference is done by performing pair-wise combinations of CPTs using multiplication (and marginalization, when appropriate) to obtain a distribution over query variables. The factoring problem is to perform these pair-wise combinations in the order that minimizes the number of floating-point multiplications. While similar to the methods described here (i.e. each internal node basically represents combinations of functions), there are several differences from our approach. First, the inference algorithm described for OFP operates on complete functions, rather than single values. Second, OFP only performs pair-wise combinations, whereas we allow more than two children per node. While this does not affect the number of multiplications, it can be shown that restricting to pair-wise combination can result in more recursive calls, decreasing the efficiency of our algorithm. Second, the authors apply OFP across all functions in a network, whereas our restructuring takes place only at a local level (across all children of one internal node). This means that operations such as marginalization need not be considered in our restructuring. However, because of their similarity, we are able to adapt a greedy heuristic from OFP to provide a considerable speedup to this restructuring process, as discussed in Section 5.

3 Unlabeled Elimination Tree Nodes

As mentioned, there is no restriction on the number of children that an internal node in an etree can have. However, each internal node must be labeled with exactly one variable. However, if etrees were allowed to contain internal nodes with no labeling variable, it would permit a variation on their structure. For example, Figure 4(a) shows node N_F and its direct descendants from the tree in Figure 2. Figure 4(b) introduces an unlabeled node below N_F. This new node now parents two of *Fire*'s children from the previous model. *Fire* now parents the new node and its old node *Smoke*. Note that the modified tree still maintains the properties of the etree set out in the previous section.

It would be fairly straightforward to modify the etree algorithm with a special case to handle nodes with no labeling variable. However, rather than modify the algorithm, we introduce a *dummy variable* to label all nodes with no labeling variable. The dummy variable, unlike the other variables, does not correspond to a variable in the Bayesian network, so its value has no effect on the values obtained from the CPTs during inference. Furthermore, the dummy variable is always "observed", that is, its value is never unknown. Labeling each empty node in this manner eliminates the need for modifications to the original algorithm.

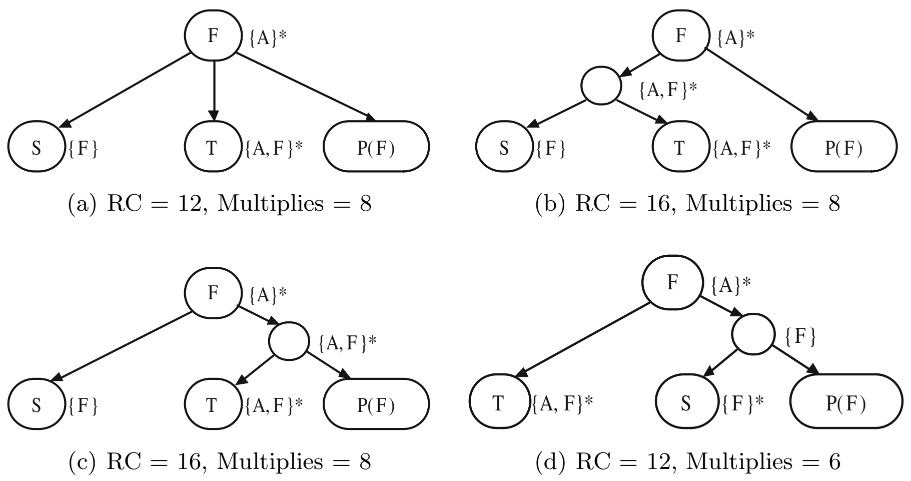

F
{A}*
S
{F}
T
{A,F}*
P(F)
{A,F}*
(a) RC = 12, Multiplies = 8
(b) RC = 16, Multiplies = 8
(c) RC = 16, Multiplies = 8
{A, F}*
{F}*
(d) RC = 12, Multiplies = 6

Fig. 4. Candidate structures for the children below F in the etree shown in Figure 2. Each structure is labeled with the number of recursive calls (RC) and multiplications that will be incurred during inference.

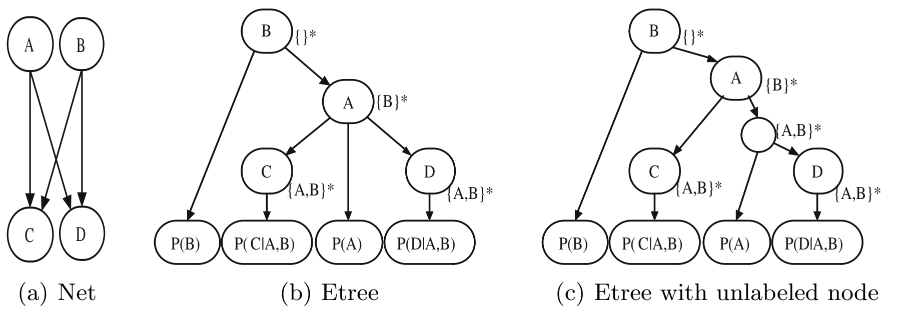


Fig. 5. An example network and corresponding etrees. In this example, the structure for N_A's children cannot be improved using unlabeled nodes.

We now demonstrate how restructuring affects algorithmic performance. The number of recursive calls made by N_F decreases by four, and the number of recursive calls at the new node is eight, so the overall number of recursive calls increases by four. Therefore, this restructuring of the tree has increased overhead by increasing the number of recursive calls, and is therefore undesirable. However, consider a different modification of the original etree, shown in Figure 4(d). In this case, the empty node now parents two different previous children of *Fire*. It can be easily verified that the number of recursive calls is not affected by this modification. However, the overall number of multiplications required is reduced by two. Furthermore, if the variables in the example network had larger domains, the restructuring of Figure 4(d) would decrease both the number of recursive calls and the number of multiplications required.

Note that it is not always possible to restructure the children of a node to reduce cost: in some cases, a flat model with no unlabeled nodes is the best one. Consider the network in Figure 5(a), and a possible etree for this network in Figure 5(b). There is no restructuring at node N_B (such as the one shown in Figure 5(c)) that will result in a reduction in the number of recursive calls *or* multiplications required by inference. For example, the tree in Figure 5(c) requires the same number of multiplications as the one in Figure 5(b), but requires more recursive calls, hence, the latter should be preferred.

4 Performing a Decomposition

Given the previous discussion, our goal is to restructure an existing etree using unlabeled nodes, in order to reduce the runtime required by inference. The restructuring process is applied individually to each node having more than two children. Each node is restructured such that it minimizes the number of recursive calls made, breaking ties using number of multiplications. The new configuration replaces the original structure. For example, when restructuring the etree in Figure 2, we consider restructuring below N_F, as it has three children. The candidate structures are shown in Figure 4. Each candidate is annotated with the number of recursive calls and multiplications that it requires.

Note that determining the best model for each node is a time consuming process. However, the restructuring process is a compile-time step. That is, restructuring is performed offline, where its cost can be amortized over many runs of the inference algorithm. However, when the number of children at a node is sufficiently large, an exhaustive search over all possible structures may be infeasible. In the next section, we introduce a fast method for restructuring, at the price of optimality.

While the restructuring process will typically reduce the number of children at each node, the resulting structure will not necessarily be binary. Figure 5 demonstrates this, as the final structure will be Figure 5(b), where node N_A has three children.

We must consider what happens to the memory requirements of the etree caches when the tree undergoes restructuring. In some cases, the memory is not affected. Consider the etrees in Figures 4(b) and 4(d). Comparing these trees to the original etree in Figure 2, we see that the memory requirements are identical, once the dead caches are pruned. However, this is not guaranteed to be the case. Figure 6(a) shows an etree node with three children. Restructuring this node (Figure 6(b)) reduces the number of multiplications by 25%, while maintaining the same number of recursive calls. However, we now need to store the cache at the unlabeled node. In the worst case, the restructuring process has the potential to increase the number of caches at each node N by $ch_N - 2$. However, the process can also *decrease* the required cache memory. Figure 7 shows an example. Figure 7(a) shows an etree node with three children prior to being restructured. Notice that all children have caches that are not dead. In Figure 7(b), the number of caches has increased, but the new cache renders two of the child caches dead. Hence, we are now storing three caches instead of four.

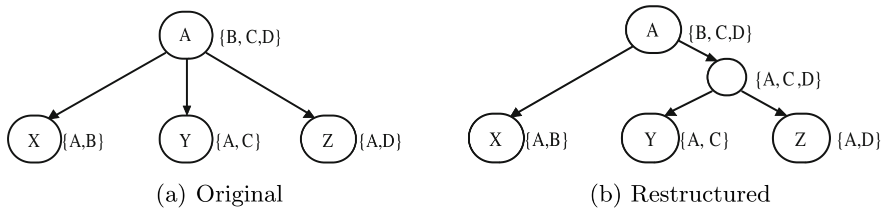


Fig. 6. A restructuring that reduces runtime, but increases memory. (a) shows a partial elimination tree, and (b) shows that tree, restructured using an unlabeled node.

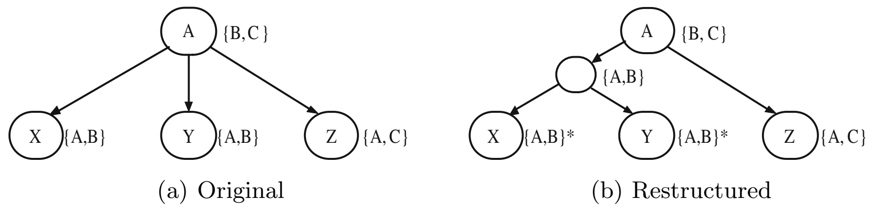


Fig. 7. A restructuring that reduces both runtime and memory. (a) shows a partial elimination tree, and (b) shows that tree, restructured using an unlabeled node.

In our test networks, the cache space typically increased, but by a reasonable amount (8% on average, and $< 24\%$ in all cases). However, in situations where an increase in space is not acceptable, the process could easily be modified to ignore any restructurings that result in an increase in memory.

We evaluated these restructured etrees using 9 well-known benchmark networks.[2] Each network was queried using no evidence, and no variable pruning (e.g. barren variables) or local structure optimizations were applied (this forces the inference algorithm to compute over the entire search space of the problem). The etrees were first compiled into a conditioning graph [6], and used the indexing structure described in [5], to reduce overhead. All tests were performed on a computer running at 2.66GHz with 2 GB of memory. Table 1 shows some measurements from our experiments. For each network, the table shows the number of recursive calls, number of multiplications, runtime, and cache memory usage using the original etree and the restructured etree. The memory usage assumes that each floating point value uses 4 bytes of memory.

The table shows several interesting points. First, the restructuring of the etrees can have a significant effect on the amount of computation done by the algorithm. The number of recursive calls was reduced in all networks, in some cases by over 50%. The number of multiplications were reduced as well, in some cases by over 70%. The overall runtime of the algorithm was reduced as well, although the decrease was less by percentage than recursive calls and multiplications. This is not surprising, as recursive calls and multiplications represent only a subset of

[2] http://www.cs.huji.ac.il/~galel/Repository/

Table 1. Experimental results. Table shows recursive calls ($\times 10^6$), multiplications ($\times 10^6$), runtime (s), and memory (MB).

Network	**RC** ($\times 10^6$)		**FM** ($\times 10^6$)		**Runtime (s)**		**Mem (MB)**	
	Old	**New**	**Old**	**New**	**Old**	**New**	**Old**	**New**
Barley	63.9	44.5	44.6	24.9	12.3	9.00	5.25	5.56
Diabetes	35.5	33.7	18.4	16.5	7.31	7.00	3.17	3.62
Mildew	19.9	19.6	10.1	9.82	3.94	3.89	0.68	0.71
Munin1	986	431	774	216	186	88.6	58.4	58.4
Munin2	12.7	10.8	7.38	5.35	2.56	2.21	2.30	2.37
Munin3	13.2	6.98	9.99	3.53	2.52	1.44	1.82	1.43
Munin4	54.8	34.6	39.6	18.8	10.6	7.16	6.90	8.49
Pigs	4.10	1.84	3.22	0.92	0.82	0.43	0.47	0.57
Water	11.6	8.93	7.05	4.34	2.35	1.90	0.98	1.19

the total operations that are going on; other significant operations, such as CPT indexing [5], are not necessarily reduced by the restructuring.

Secondly, we can see that restructuring the tree generally results in an increase in memory usage. However, the memory increase was relatively small, averaging 8% over our test cases, and never exceeding 24%. As mentioned, the restructuring has the potential to also decrease memory usage. We can see that in Munin3, the overall cache memory usage was actually *decreased* by 21%.

Note that the cost-benefit ratio is not consistent across all networks. For example, there is only a 5% reduction in the number of recursive calls for the *Diabetes* network, at the cost of a 14% increase in cache memory. However, remember that these values can be computed offline, during tree construction. Hence, such cost-benefit analysis can be made prior to runtime, allowing the user to decide whether to use the original or reconstructed etree.

While these tests show the effect that restructuring has on overall inference, they do not demonstrate the per-node savings in each network. Define the *recursive call ratio* of node N to be the ratio between the number of recursive calls that N requires after restructuring (which includes the extra recursive calls made to and by the unlabeled nodes between N and its children), and those required by N prior to restructuring. The *multiplication ratio* of node N is defined similarly, replacing recursive calls with floating point multiplications in the definition. Table 2 shows the maximum ratios across all nodes for each network.

The table shows several points. First, even in networks whose overall performance was only affected slightly by restructuring (e.g. Diabetes, Mildew), there existed nodes whose restructuring reduced the number of necessary multiplications by nearly 50%. Furthermore, even in networks where restructuring did have a substantial impact on overall inference, the per-node savings are even more dramatic. A particularly striking example is in the Pigs network, where the restructuring of one of its etree nodes reduced the number of recursive calls at that node by over 75%, and multiplications by over 87%.

Table 2. Maximum ratios between original and restructured nodes, in terms of recursive calls (RC) and floating-point multiplications (FM)

	Barley	Diabetes	Mildew	Munin1	Munin2	Munin3	Munin4	Pigs	Water
RC	0.34	0.68	0.69	0.25	0.26	0.26	0.25	0.23	0.50
FM	0.20	0.51	0.51	0.15	0.15	0.15	0.14	0.12	0.34

5 Fast Restructuring

When restructuring the children of a particular node, the algorithm described in the previous section searches exhaustively through each possible structure. It is easily shown that the number of ways to restructure the children of a node grows exponentially with the number of children. Considering our particular implementation, the algorithm was quite fast (< 0.1 s) for nodes with 7 children or less. However, restructuring a node with 12 children took roughly 10 minutes, and a 14 child node took over 10 hours. Hence, for networks whose elimination tree contains nodes with many children, an exhaustive search may be infeasible. In this section, we consider a greedy approach to restructuring. Our approach is based on a heuristic from the *optimal factoring problem* [9], summarized in Section 2.

A rough outline of the algorithm is as follows. Each node N is restructured bottom-up, beginning with its children. At each iteration, we choose two nodes whose combined cache-domains have the smallest weight. These two nodes are removed from the pool of nodes, and become the children of a new unlabeled node, which is then added back to the pool. Unlabeled nodes whose cache-domains are the same are merged. Unlabeled nodes whose cache-domains are equivalent to the cache-domain of N plus the labeling variable of N are absorbed by node N (i.e. the unlabeled node is eliminated, and N adopts its children).

Table 3. Experimental results, comparing trees constructed exhaustively (Ex) and using the greedy heuristic (Heu). Note that RT=restructuring time. Runtimes requiring less than 0.01s are denoted by a dash.

Network	**RC (x10^6)**		**Runtime(s)**		**Mem(MB)**		**RT (s)**	
	Ex	**Heu**	**Ex**	**Heu**	**Ex**	**Heu**	**Ex**	**Heu**
Barley	44.5	44.5	9.00	9.01	5.56	5.56	0.02	0.01
Diabetes	33.7	33.7	7.00	6.99	3.62	3.62	0.01	0.01
Mildew	19.6	19.6	3.89	3.88	0.71	0.71	–	–
Munin1	431	431	88.6	88.8	58.4	58.4	0.37	0.08
Munin2	10.8	10.8	2.21	2.21	2.37	2.38	1.21	0.02
Munin3	6.98	7.01	1.44	1.44	1.43	1.43	13.4	0.02
Munin4	34.6	34.6	7.16	7.17	8.49	8.49	1.85	0.04
Pigs	1.84	1.84	0.43	0.43	0.57	0.57	1.66	0.01
Water	8.93	8.93	1.90	1.91	1.19	1.19	0.01	–

This heuristic effectively solves the restructuring time issue, with nodes having 100 children restructuring in less than half a second. The quality of the restructurings was examined over our test networks; the results of these tests are shown in Table 3. As the table demonstrates, not only does the heuristic provide a considerable speedup in restructuring times, but the trees resulting from fast restructuring performed nearly as well as the exhaustive search algorithm in all circumstances.

6 Conclusion

In this paper, we consider alternate construction methods for etrees, that are designed to reduce runtime when performing inference. Specifically, we relax the constraint that each internal node must be labeled by a variable from its corresponding Bayesian network, and allow *unlabeled nodes*. Through the use of these unlabeled nodes, we demonstrated that the amount of computation required by these structures can be reduced considerably: over a 55% reduction in recursive calls and a 70% reduction in multiplications over some of our test cases. This optimization occurs with no effect to the variable elimination ordering, and requires no modification to the actual inference algorithm; furthermore, it incurs a relatively small memory cost. We also demonstrated a greedy approach to restructuring, that's based on the optimal factoring problem, which gives nearly identical results to the exhaustive search, while improving the restructuring time by orders of magnitude for nodes with many children. Although these techniques were showed in the context of conditioning graphs, they could easily be applied to other recursive decompositions (e.g. dtrees).

We are currently considering several extensions to this project. As an example, restructuring the etree can result in an increase in cache memory requirements. While the observed increases were quite reasonable in our test cases, there may be networks for which memory increases may be unacceptable. In these situations, one would still like to try to apply this restructuring optimization as much as possible, while using (for example) the current cache sizes as an upper bound on memory usage. One trivial approach would be to perform a tree restructuring, and reject it if the resulting tree requires more memory. A more refined approach would be to eliminate any restructurings that resulted in a memory increase, but on a node-by-node basis. Finally, one could imagine an interplay between nodes, where some nodes would be allowed to increase their memory usage if other nodes decreased theirs.

Acknowledgements

Thanks to the anonymous reviewers for their suggestions regarding the paper. A special thanks goes out to Michael Horsch for his helpful comments. This research is funded by NSERC; their support is gratefully acknowledged.

References

1. Allen, D., Darwiche, A.: New Advances In Inference By Recursive Conditioning. In: Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence, pp. 2–10 (2003)
2. Cooper, G.F.: The Computational Complexity of Probabilistic Inference using Bayesian Belief Networks. Artificial Intelligence 42, 393–405 (1990)
3. Darwiche, A.: Recursive Conditioning: Any-space Conditioning algorithm with Treewidth-bounded Complexity. Artificial Intelligence, 5–41 (2000)
4. Darwiche, A., Hopkins, M.: Using Recursive Decomposition to Construct Elimination Orders, Jointrees and Dtrees. In: Benferhat, S., Besnard, P. (eds.) ECSQARU 2001. LNCS (LNAI), vol. 2143, pp. 180–191. Springer, Heidelberg (2001)
5. Grant, K.: Efficient Indexing for Recursive Conditioning Algorithms. In: Proceedings of the the 23rd International FLAIRS Conference, pp. 537–542 (2010)
6. Grant, K., Horsch, M.: Conditioning Graphs: Practical Structures for Inference in Bayesian Networks. In: Proceedings of the The 18th Australian Joint Conference on Artificial Intelligence, pp. 49–59 (2005)
7. Grant, K., Horsch, M.C.: Methods for Constructing Balanced Elimination Trees and other Recursive Decompositions. International Journal of Approximate Reasoning 50(9), 1416–1424 (2009)
8. Lauritzen, S., Spiegelhalter, D.: Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. J. Royal Statistical Society 50, 157–224 (1988)
9. Li, Z., D'Ambrosio, B.: Efficient Inference in Bayes Networks as a Combinatorial Optimization Problem. Int. J. Approx. Reasoning 11(1), 55–81 (1994)
10. Monti, S., Cooper, G.F.: Bounded Recursive Decomposition: a Search-based Method for Belief-network Inference under Limited Resources. International Journal of Approximate Reasoning 15(1), 49–75 (1996)
11. Poole, D., Mackworth, A., Goebel, R.: Computational Intelligence. Oxford University Press, Oxford (1998)
12. Zhang, N., Poole, D.: A Simple Approach to Bayesian Network Computations. In: Proceedings of the 10th Canadian Conference on Artificial Intelligence, pp. 171–178 (1994)

CBR and Neural Networks Based Technique for Predictive Prefetching

Sohail Sarwar[1], Zia Ul-Qayyum[2], and Owais Ahmed Malik[1]

[1] School of Electrical Engineering and Computer Sciences,
National University of Sciences and Technology, Islamabad, Pakistan
[2] University Institute of Information Technology, University of Arid Agriculture
Rawalpindi Pakistan
sohail.sarwar@seecs.edu.pk, ziaqayyum@uaar.edu.pk,
owais.malik@seecs.edu.pk

Abstract. Cache prefetching in memory management greatly relies upon effectiveness of prediction mechanism to fully exploit available resources and for avoiding page faults. Plenty of techniques are available to devise strong prediction mechanism for prefetching but they either are situation specific (Locality of reference principle) or inadaptable (Markovian model) and costly. We have proposed a generic and adaptable technique benefiting from past experience by employing hybrid of Case Based Reasoning (CBR) and Neural Networks (NNs). Here we will be concerned with improving adaptation phase of CBR using NN and its impact on predictive accuracy for prefetching. The level of predictive accuracy attained (specifically in case adaptation of CBR) is ameliorated by handsome margin with declined cost than contemporary techniques as would be affirmed by results.

Keywords: Prefetching, Case Based Reasoning, Neural Networks, Locality of Reference, Markovian Model.

1 Introduction

It is almost inevitable to live without information systems after revolutionary advent of Information Technology (IT) employed somehow in every aspect of human life. These systems are exploited in various I/O intensive applications with a broader scope overarching business, industry, academia, medical sciences etc. For operation of these information systems, the required level of storage and computing resource varies depending upon the volume of data and compute-intensiveness of applications while manipulating bulk of information. In previous decade, we can observe not only massive increase in capacity of storage media and computing resources but their diminished costs can't go unnoticed as well.

Despite the important role of Information Systems in human life, the issue of disparity between 'execution power of processors' and 'performance of peripherals' (esp. Hard Disk Drives) is not new to the research community. With each day, this performance gap is widened because of delays involved in the form of seek time, rotational delay and block transfer time while manipulating the data. The more alarming aspect is per year improvement where CPU speed improves by 50% each year and disk performance improves just by 8% a year [1].

We can observe a lot of efforts [2, 3, 4] made to match I/O performance of these peripherals to the speed of processors ending up in notable success. This improvement is courtesy to some available “cache prefetching” techniques with their respective pros and cons [2]. Prefetching is speculatively fetching data that will be accessed in future for preventing wastage of CPU cycles, searching from slow Hard Disk Drives (HDDs) and hence concealing IO latencies [3]. So the ultimate purpose of prefetching is to avoid the conditions of page fault where demanded page has to be accessed from HDDs and processor remains in stall state until data is fetched in main memory.

We can assert prefetching as process of visualizing the future requests so a strong prediction mechanism is required of prefetching right data before hand thus improving system performance by enhancing “ hit ratio of cache”. This requirement of right prediction has been acclaimed using various techniques such as ‘Locality of Reference Principle [3]’, ‘Markovian Model [5]’ and ‘Neural Networks [4]’ to mention a few.

Locality of reference principle is congregation of two heuristics named ‘Temporal Locality’ and ‘Spatial Locality’. ‘Temporal locality’ is based on assumption that pages that have been referenced now are more likely to be re-referenced in near future, so they must be retained in the memory (primary). Other principle is based on the probability of demanding the pages that lie in a sequence next to the currently accessed pages, so pages residing in next sequence are fetched in advance to exploit the benefits of prefetching. We can imply that both the heuristic have fundamental role in prefetching but are situation specific i.e. first heuristic is unable to deal with random requests and other one specifies no time for retaining the pages.

Markov Prediction algorithm [5] has also been a strong candidate for its implementation in prefetching techniques. The last part of reference stream is compared with prefixes of previous reference streams. The remaining patterns are prefetched in memory. So the success of algorithm depends upon patterns in data stream. In addition to this dependence Markov algorithm does not adapt itself to the situation for decision make.

In [4], Neural Network based model has been used to predict suitable candidate for prefetching but the technique has not been much effective in terms of prediction as well as cost (in terms of mean square error). The issues arose due to usage of raw data for experimentation and lack of appropriate knowledge base that could benefit the past experience.

Considering these issues with prevalent techniques, we come up with hybrid of two Artificial Intelligence (AI) techniques named ‘Case Based Reasoning’ [4] and ‘Neural Networks’ [5] for improvising level of predictive accuracy in finding appropriate prefetch-candidates. We have tried to benefit the previous experience by having solution for newly emerging problems from CBR’s case base and on the other side utilized the adaptive nature of NN for problems having similar but inexact solutions in case repository of CBR (i.e. for case adaptation).

CBR’s case adaptation is carried out through Rule Base Systems (RBSs) most of the time as observed during literature survey. But there are few short falls in RBS as given in [6]. Considering the concerns posed while using set of rules for CBR case adaptation, we work out this phase of CBR via NN (in hybrid of CBR and NN). The proposed technique enhances the level of predictive accuracy compared to contemporary techniques used for case adaptation.

Few of these issues with RBS are given below [6]:

- Dependent and independent attributes can't be defined by user.
- Even if attributes are defined, non-linear relation (as in our case) cannot be defined as IF then Else.
- In order to devise the rule base system, user must have complete knowledge of underlying data to devise the assertions.
- More intricate process when multiple case bases are used.
- User must maintain rule base to be consistent with incremental growth of case base.

In this paper resolution to some of the above issues is proposed. Different datasets (Integer strings) of different sizes have been used for carrying out different experiments. These datasets are furnished in the CBR's case base in form of cases each of length 9 (see section 3.1 for details). On the basis of first eight elements 9th element is predicted (as a candidate to be prefetched). Detailed view of experimental setup is available in section 3.

Section 2 discusses architecture of proposed system, while Section 3 is furnished with implementation details. Section 4 contains the results and discussion on experiments and comparison of an existing technique with the proposed one. Section 5 concludes the work and reveals intended future direction.

2 System Architecture

The architecture of proposed system is presented in Figure 1. It has four main modules namely Preprocessor module, Case retrieval, Case Reuse and Case Adaptation.

Dataset Preprocessing Module: This module is concerned with populating the CBR's case base (or System's Case Base). There has been an issue with representation of data due to randomness or absence of rich feature set for pattern extraction which will later be used. In order to address the mentioned limitation the preprocessing activity is incorporated in system that takes random sized Hex strings (each representing memory address) at constant temporal scale. The address strings plotted in a Suffix Trees [7] are generated using these address strings while keeping track of frequencies of each of the addresses in the address string. Later, these branches based on their occurrence frequency (repetitiveness) are extracted and placed in the CBR's case base.

Case Retrieval module: This module provides query specific solution using cases from the case base i.e. for new address string (query). The algorithm in this module computes level of similarity of query string vs cases in the case base. This similarity is computed using uses 'Tversky Ratio model [10]' as a similarity metric. If computed similarity is 100%, that is 'n-1' elements of two cases are same then the solution of that problem will be selected and applied.

Conversely, the selected case which is similar to certain threshold (varied between 60%-90%) is subjected to the *Case Adaptation module* for getting nearest but probable solution.

Case Adaptation module: The adaptation is carried out using case adaptation module in two ways i.e. 'Majority Class Voting' and 'Neural Networks' respectively (details in coming section). After adaptation is carried out, the adapted case will be reused and retained to the case base for future usage.

Case Reuse module: This module has been discussed to make this paper self-contained; though results of this component are not part of this work.

3 Implementation Strategy

In this section, the information details of proposed system are discussed.

3.1 Data Preprocessing

The core issue for building CBR model is to devise some technique for populating the case repository of CBR. In our system, this activity is carried out using Suffix Tries [8] where each random sized address string in Hex format is read over a constant temporal scale (reading each string for few micro-seconds). These Hex strings are converted to equivalent decimal for plotting address nodes on Suffix Trees. Each node in our suffix tree contains three parts: data part, frequency-counter and a flag.

Data part of node contains actual address; frequency-counter refers to how many times that data occurred in a branch of suffices derived from address strings and flag is used to distinguish intermediary nodes from leaf nodes.

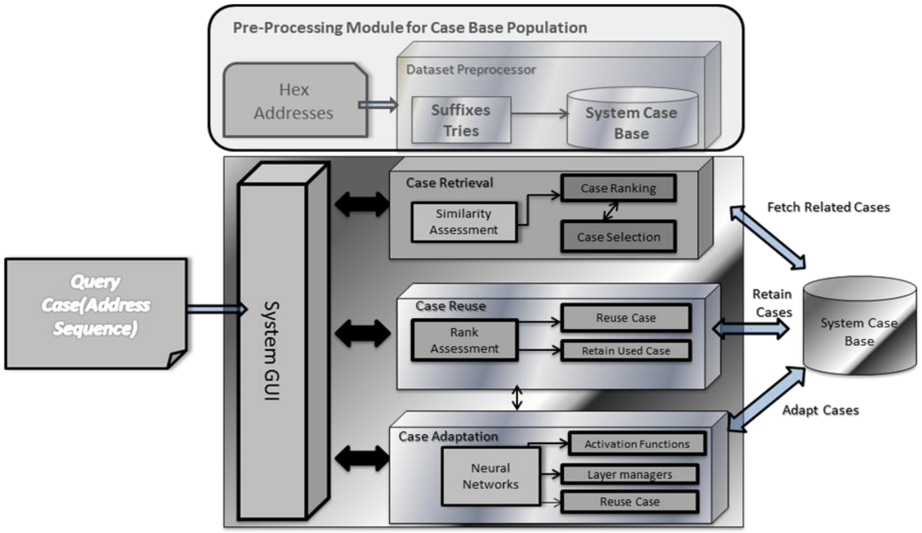


Fig. 1. Architecture of the proposed system

For example, we get the tree in the form of Figure 3 on plotting the following two strings with their respective suffices:

String 1	356982
String 2	569249

Whenever an occurrence of a suffix is repeated, the frequency of respective node is incremented by one. After all the address strings are plotted by "Preprocessor", the next step is extraction of the repetitive patterns (i.e. branches of tree plotted); this extraction was done using Depth First Search (DFS). We extracted all the branches of length 9 with varying frequency depending upon number of cases yielded during branch extraction. Each of these branches of length n=9 was used to represent a case of CBR's case base. According to our heuristic 'n-1' elements would refer to the problem attributes and nth element is considered as respective solution.

In other words, a combination of first eight addresses refers to prediction of 9th address. This situation can better be apprehended from Figure 2.

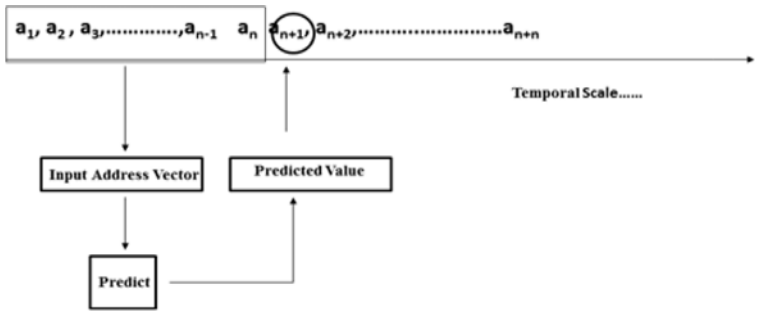

$a_1, a_2, a_3, \ldots, a_{n-1}$ a_n $a_{n+1}, a_{n+2}, \ldots a_{n+n}$
Temporal Scale......
Input Address Vector
Predicted Value
Predict

Fig. 2. Representation of a Case in CBR Case

Once the case repository is populated, the CBR is ready for its operative tasks of retrieval and adaptation. However, There might be a question why length 9? We experimented different lengths of cases in CBR's case base and observed an added cost in retrieval as well as adaptation using NN when length was kept more than 9. Reducing the length ended up in loss of predictive accuracy. So 9 was the lucky number exhibiting an acceptable tradeoff between the cost and accuracy (For practical purpose it depends on the page and cache size).

3.2 Case Retrieval

Case retrieval is carried out by measuring the level of similarity among query cases and cases in case base. We used the similarity measures based on "Ratio Model" by Tversky [5] while performing similarity assessment in CBR:

$$\text{SMpq} = \frac{\#\text{of matched Addressed cases for Query Case}}{\#\text{of matches in CB} + \#\text{of difference s in CB}}$$

Once similarity assessment is complete, each case is assigned a rank in order to predict the level to which query case is similar to the cases in case base.

Here, it's worth mentioning that there is plethora of similarity metrics available but this is the only metrics that computes the 'rank' of similarity while preserving sequence and semantics of each case. One of the other options was using Fuzzy Logic [13] requiring clustering of dataset based upon feature weights. We neither had rich feature set nor the respective weights impeding the use of Fuzzy Logic based metric. The metric based on Euclidian distance provides the difference between the corresponding attributes of problem case and query cases which was not workable in our model as we were more interested in finding the commonalities among the cases rather than computing the difference. So we found ratio model as an appropriate solution for our model.

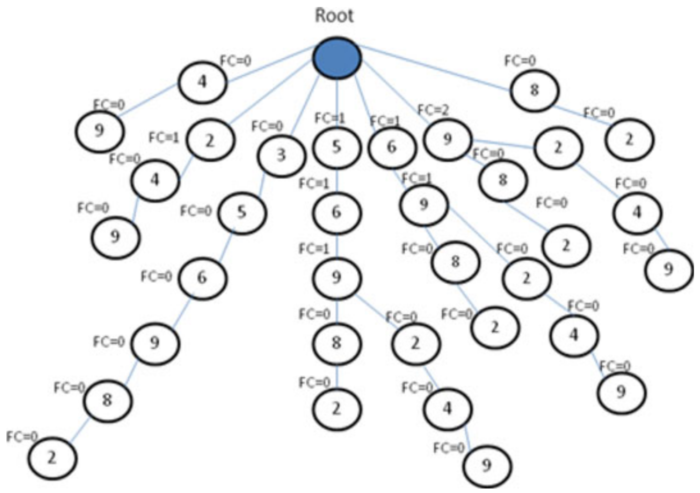


Fig. 3. Suffix tree 356982 and 569249

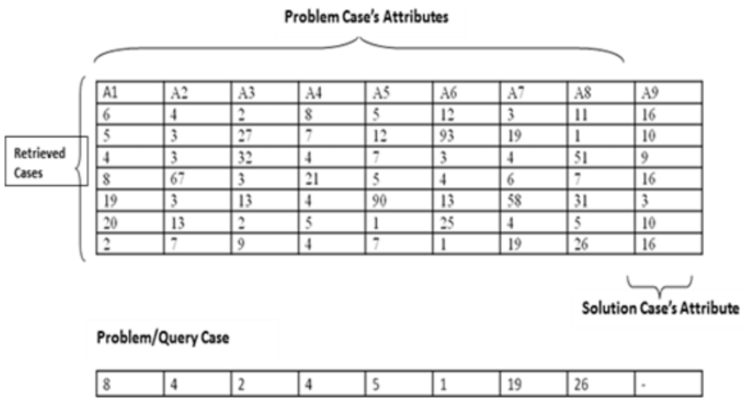
3.3 Case Adaptation of CBR Cases

After similarity assessment of new case, we make a decision about case adaptation in order to reach an appropriate solution if we have no similar problem and respective solution. The situation in which we have to go for adaptation is illustrated in the Figure 4. We retrieve a case that has different problem attributes (A1 and A3) and we need to predict the appropriate solution attribute (A9) based on adaptation technique.

Adaptation was performed in two ways (1) Majority Class Voting and (2) Using Neural Networks. These two techniques were compared to find their respective impact on predictive accuracy in prefetching. We will describe each of the approaches briefly.

3.3.1 Majority Class Voting (MCV) Based Adaptation

The concept of majority class voting is used to perform adaptation while exploiting the frequency of solution attribute from retrieved cases. In other words, we classify the retrieved cases with respect to the last element of retrieved cases. We keep the track of solution (9th element) for our problem case (first eight attributes) over the set of retrieved cases. On the basis of solution element, we classify the cases. Then probability of each class is computed for retrieved cases. The class with highest probability will be most probable solution for our problem. In other words, the class containing maximum number of cases (in terms of solution attribute) is considered to be most appropriate solution for query case.


Problem Case's Attributes
Retrieved Cases
Solution Case's Attribute
Problem/Query Case

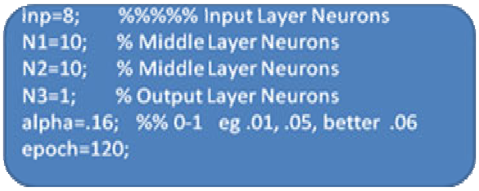
A1	A2	A3	A4	A5	A6	A7	A8	A9
6	4	2	8	5	12	3	11	16
5	3	27	7	12	93	19	1	10
4	3	32	4	7	3	4	51	9
8	67	3	21	5	4	6	7	16
19	3	13	4	90	13	58	31	3
20	13	2	5	1	25	4	5	10
2	7	9	4	7	1	19	26	16

8	4	2	4	5	1	19	26	-

Fig. 4. Case adaptation using MCV

3.3.2. Neural Networks Based Adaptation

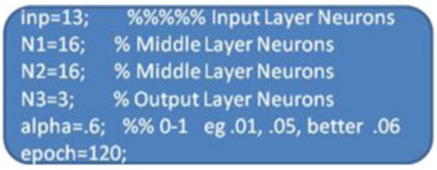
The other task is to devise the computation model of Neural Networks for determining the predictive accuracy of NN in case adaptation. For the purpose of training and validation, we used the cases of CBR. We used Back Propagation Neural Networks with following configurations in Matlab7.



```
inp=8;      %%%%% Input Layer Neurons
N1=10;      % Middle Layer Neurons
N2=10;      % Middle Layer Neurons
N3=1;       % Output Layer Neurons
alpha=.16;  %% 0-1   eg .01, .05, better .06
epoch=120;
```

These are the configuration parameters for the layers of NN which are based on the experiments performed to optimize the results. Since the problem case has eight attributes, so the input layer has eight neurons and one output is modeled by one neuron.

We have described how CBR and NN work in their entirety with respect to operation of predictive accuracy. In order to evaluate hybrid of CBR and NN, training of neural network was performed on retrieved cases only. These cases were retrieved using metrics described in section 3.2. The validation set comprised of the query case with inexact match. We further evaluated our hybrid system to one of the existing systems [4] for predictive prefetching using neural networks only on following parameters (these parameters were used by author of paper and we address his approach as VANN).



```
inp=13;      %%%%% Input Layer Neurons
N1=16;     % Middle Layer Neurons
N2=16;     % Middle Layer Neurons
N3=3;      % Output Layer Neurons
alpha=.6;  %% 0-1  eg .01, .05, better .06
epoch=120;
```

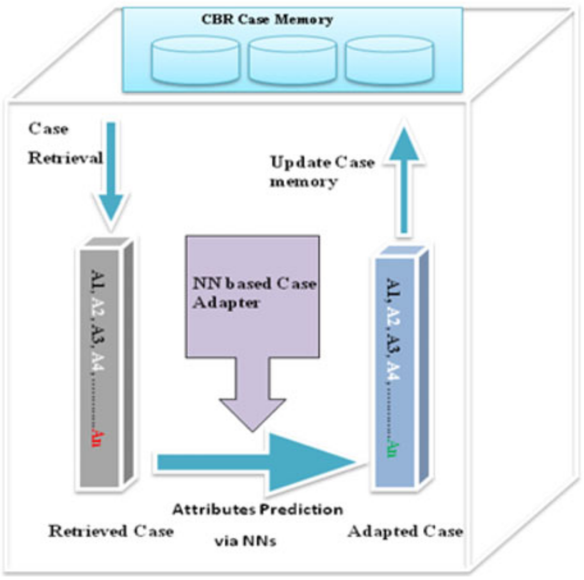


Fig. 5. Case adaptation for page prediction

The hybrid of system showed better performance than all contemporary systems as would be depicted in next section.

4 Results and Evaluations

We present the results based on observations made during the adaptation phase as mentioned in section 3.3.

The comparison of two techniques is presented followed by a comparative analysis of proposed technique with an existing technique named 'VANN [4]'. We used the same NN model and parameters as given in [4] for the comparison purposes.

There were 1200 cases in our case base initially. Random query cases were given to retrieve the cases using our similarity metric discussed in 4.2 from the case base in the form of different sets (Each set carrying 100 cases). Firstly, the retrieved cases with similarity not equal to 100% were adapted using Majority Class Voting (MCV). We measured percent similarity to sort out most probable solution using MCV.

On the other side, to evaluate the case of CBR+NN, NN model was trained on sets of the retrieved cases. The retrieval of these cases was based upon CBR's similarity metric and adaptation was carried out using Feed Forward Back Propagation Neural Networks. Training set (retrieved cases) and validation sets contained 600 and 800 cases respectively. For the purpose of validation, we gave right cases in validation set to test the accuracy towards the required solution.

We performed a comparative analysis of our technique with one of the existing approaches as asserted above, where we selected most frequently occurring branches from Suffix tree (frequency <=10) for CBR's case base unlike previous training set with less frequent data patterns (frequency <=4). The model presented in [7] can hardly be compared to proposed model of 'CBR+NN' and none of the approaches can be compared to hybrid of CBR and Neural Networks.

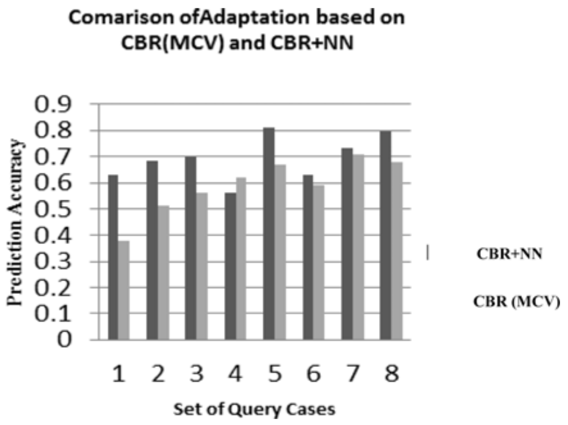


Fig. 6. Case Adaptation performance using CBR (MCV) vs. CBR+NN

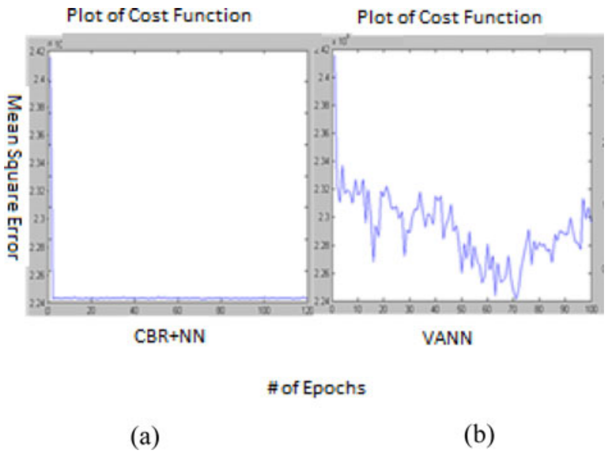


Fig. 7. Mean Square Error Cost model of "CBR+NN (a)" vs "VANN (b)"

This fact can be viewed from Figures 6, 7, 8 and Table 2. The predictive performance of proposed 'CBR+NN' when compared to performance of CBR (using Majority Class Voting concept in 3.2) 'NN' (section 3.3) and to VANN, has outperformed these techniques not only in terms of predictive accuracy but the cost associated was also minimal to a satisfactory level.

Although the Mean Square Error (MSE) cost seems increasing exponentially at initial stage in both the cases (CBR+NN and VANN) but proposed model (Fig. 7a) would converge towards zero gradually contrary to cost of VANN (Fig. 7b) where one can observe not only inconsistent behavior since beginning but after 73rd iteration it shoots up again.

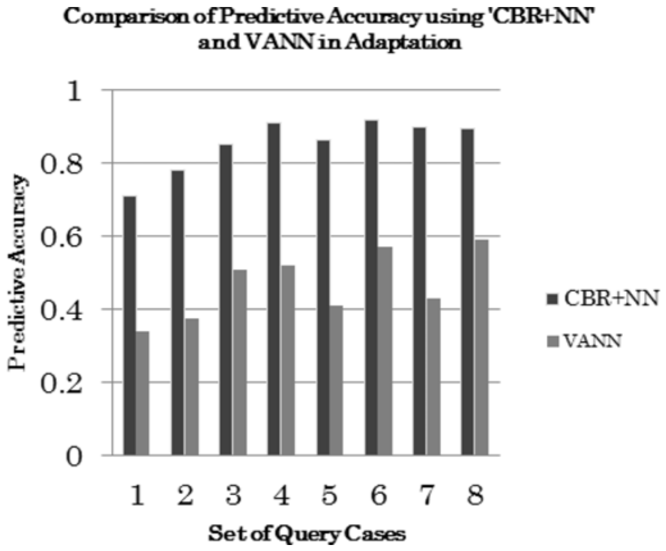


Fig. 8. Predictive Accuracy of CBR+NN vs. VANN

All these experiments are also represented in the form of percent "Precision and Recall". Standard precision and recall measures are computed as follows:

$$Recall = \frac{|\, Relevant\ Cases \cap Retrieved\ Cases \,|}{|\, Relevant\ Cases \,|},$$

$$Precision = \frac{|\, RelevantCases \cap RetrievedCases \,|}{|\, RetrievedCases \,|},$$

$$F-measure = \frac{2 * Precision * Recall}{Precision + Recall},$$

F-measure is a single measure that trades off Precision versus Recall [15]. We have taken average of results illustrated in the Figure 6 and Figure 8 and computed average precision and recall using above formulas. Handsome amount of precision is being achieved using our technique which further strengthens effectiveness of proposed idea as shown in Table 1. Apart from precision, recall also suggests that overall coverage of getting results is also well comprehended.

Table 1. Comparison in terms of Precision & Recall

Techniques	Precision	Recall	F-measure
CBR+ANN	78%	90%	83.5%
CBR(MCV)	59%	78%	67%
VANN	43%	58%	49%

The results revealed that proposed approach of "CBR+NN" has better predictive performance than fellow techniques. The differentiating aspect is usage of suffix tries in preprocessing mode, which provides rich and natural data patterns out of available data. The presence of these patterns provides effective learning for neural networks and training on retrieved cases will further purify the training set and hence improves the productiveness of proposed approach.

5 Conclusion

We propose the perfecting technique for high I/O intensive system and implemented its architecture (partially) to improve the predictive-ness in prefetching. We presume that it will lower down the impact of latency and would ensure optimum resource usage via prefetching. This prefetching would be based on intelligent pattern manipulation specific to memory addresses. The novelty of approach is signified to incorporation of suffix tries for CBR's case base population, and merger of Neural Networks with CBR in extracting and recognizing knowledge patterns for case adaptation.

We look forward to evaluate the retrieval phase by devising different similarity metrics because the metrics used are very crude one. Eventually, we try to replace the BPNN by some other model such as RBF for further optimizing the solution by minimizing the training set.

References

1. Takahashi, H., Ahmad, H.F., Mori, K.: Layered Memory Architecture for High IO Intensive Information Services to Achieve Timeliness. In: HASE 2008 (2008)
2. Papathanasiou, A.E., Scott, M.L.: Aggressive Prefetching: An idea whose time has come. University of Rochester (2005), `http://www.cs.rochester.edu/~papathan,scott`
3. IO data Prefetching based on Sequential Stream Recognition, `http://www.cs.unh.edu/~verkik/publication/cache.pdf`

4. Vardan, S.V.: Application of NN in predictive prefetching. K. R. Vaishnav Shanmugha Arts Science Technology and Research Academy (2005)
5. Mowry, T.C., Lam, M.S., Gupta, A.: Design and evaluation of a compiler algorithm for prefetching. In: Proc. of Fifth Int'l Conf. on Proceedings of the fifth international conference on Architectural support for programming languages and operating systems, pp. 62–73 (October 1992)
6. Whar, S.Y., Babka, O.: Neural Network Supported Adaptation in Case based Reasoning. In: Knowledge-Based Systems Centre, School of Computing, Information System and Mathematics, South Bank University, London, UK, GB, December 01, pp. 264–276 (2001)
7. Ukkonen, E.: On–line construction of suffix trees. In: Proc. Information Processing 92. IFIP Transactions A-12, vol. 1, pp. 484–492. Elsevier, Amsterdam (2005)
8. Ukkonen, E.: Constructing Suffix Trees On-Line in Linear Time. In: Leeuwen, J.v.(ed) Algorithms, Software, Architecture. Information Processing 1992, Proc. IFIP 12th World Computer Congress, Madrid, Spain, vol. 1, pp. 484–492. Elsevier Sci. Publ., Amsterdam (1992)
9. Khan, M.U., Ch, M.Q., Ahmad, H.F., Ali, L., Ali, A., Suguri, H.: Merging CBR and Neural Networks for SLA-Based Radio Resource Management for QoS Sensitive Cellular Networks. In: ISADS-ACM, pp. 263–269 (2007) ISBN:0-7695-2804-X
10. Sankar, K.P.: Foundations of Soft Case-based Reasoning. Indian Statistical Institute Simon c. K. Shiu Hong Kong Polytechnic University. John Wiley & Sons, Chichester (2004) ISBN:0-89791-187-3-X
11. Murray, K., Pesch, D.: Neural Network based Adaptive Radio Resource Management for GSM and IS136 Evolution. In: ISSC 2002, Cork, Ireland (June 2002)
12. Pan, S., Cherng, C., Dick, K., Ladner, R.E.: Algorithms to Take Advantage of Hardware Prefetching. In: Proceedings of the Nineteenth Annual ACM Symposium on Parallel Algorithms and Architectures
13. Finnie, G., Sunt, Z.: Similarity and Metrics in Case-Based Reasoning. International Journal of Intelligent Systems 17, 273–287 (2002)
14. Wilke, W., Bergmann, R.: Techniques and knowledge used for Adaptation during Case-Based Problem Solving. In: Proceeding of 11th International Conference on Industrial and Engineering Applications of AI and ES (1998)
15. Keith, C.J., Van Rijsbergen: A new theoretical framework for information retrieval. In: Proceedings of the 9th annual international ACM SIGIR conference on Research and development in information retrieval Palazzo, Pisa, Italy, pp. 194–200 (1986) ISBN:0-89791-187-3

Combining Neural Networks Based on Dempster-Shafer Theory for Classifying Data with Imperfect Labels

Mahdi Tabassian[1,2], Reza Ghaderi[1], and Reza Ebrahimpour[2,3]

[1] Faculty of Electrical and Computer Engineering,
Babol University of Technology, Babol, Iran
r_ghaderi@nit.ac.ir
[2] School of Cognitive Sciences, Institute for Research in Fundamental Sciences (IPM),
Tehran, Iran, P.O. Box 19395-5746
{tabasian,ebrahimpour}@ipm.ir
[3] Department of Electrical and Computer Egineering,
Shahid Rajaee Teacher Training University, Tehran, Iran

Abstract. This paper addresses the supervised learning in which the class membership of training data are subject to uncertainty. This problem is tackled in the framework of the Dempster-Shafer theory. In order to properly estimate the class labels, different types of features are extracted from the data. The initial labels of the training data are ignored and by utilizing the main classes' prototypes, each training pattern, in each of the feature spaces, is reassigned to one class or a subset of the main classes based on the level of ambiguity concerning its class label. Multilayer perceptrons neural network is used as base classifier and for a given test sample, its outputs are considered as basic belief assignment. Finally, the decisions of the base classifiers are combined using Dempster's rule of combination. Experiments with artificial and real data demonstrate that considering ambiguity in class labels can provide better results than classifiers trained with imperfect labels.

Keywords: Data with imperfect labels, Dempster-Shafer theory, Classifier combination, Neural network.

1 Introduction

In the classical supervised classification framework, a classifier is trained based on a learning set in the form of labeled patterns. However, in some applications, unambiguous label assignment may be difficult, imprecise and expensive. Such situations can occur when differentiating between two or more classes is not easy due to lack of information required for specifying certain labels to data or the difficulty of labeling complicated data of the problem at hand.

Combination of multiple classifiers' decisions induced from different information sources has proved to be a promising approach for improving the performance of a classification system that deals with imprecise and uncertain information. The Dempster-Shafer (D-S) theory of evidence [1] is a well suited framework for representation of partial knowledge. Compared to the Bayesian approach, it provides a more flexible mathematical tool for dealing with imperfect information. It offers various tools for combining several items of evidence and, as understood in the transferable belief model (TBM) [2] , allows to make decision about the class of a given pattern through transforming a belief function into a probability function. Thanks to its flexible characteristics to represent different kind of knowledge, the D-S theory provides a suitable theoretical framework for combining classifiers especially those learned using imprecise and/or uncertain data [3-7].

In this paper we propose a new approach within the belief functions and combining classifiers frameworks for dealing with supervised classification problems in which the labels of the learning data are imperfect. Several representations of the data are used and an approach is suggested, based on the supervised information, for detecting inconsistencies in the labels of the learning data and assigning crisp and soft labels to them. Multilayer perceptrons (MLP) neural network is used as base classifier and its outputs are interpreted as belief function. Final decision about the class of a test pattern is made by combining the beliefs produced by the base classifiers using Dempster's rule of combination.

The paper is organized as follows. In Section 2 basic concepts of the D-S theory are reviewed. Details of our proposed method are described in Section 3. It is followed by the experimental results on artificial and real data in Section 4 and finally, Section 5 draws conclusion and summarizes the paper.

2 Dempster-Shafer Theory

The Dempster-Shafer (D-S) theory of evidence [1] is a theoretical framework for reasoning with uncertain and partial information. Several models for uncertain reasoning has been proposed based on the D-S theory. An example is the transferable belief model (TBM) proposed by Smets [2]. In this section, the main notions of the D-S theory are briefly reviewed.

Let $\Omega = \{\omega_1,...,\omega_M\}$ be a finite set of mutually exclusive and exhaustive hypotheses called the *frame of discernment*. A *basic belief assignment* (BBA) is a function $m: 2^\Omega \rightarrow [0,1]$ verifying $\sum_{A\subseteq\Omega} m(A) = 1$. A BBA m such that $m(\phi) = 0$ is called *normal*. The subsets A of Ω with nonzero masses are called the *focal elements* of m and *m(A)* indicates the degree of belief that is assigned to the exact set of A and not to any of its subsets. There are *belief* and *plausibility* functions associated with a BBA and are defined respectively, as follow:

$$\begin{cases} Bel(A) = \sum_{B\subseteq A} m(B) & (1) \\ Pl(A) = \sum_{A\cap B\neq\phi} m(B) & (2) \end{cases}$$

Bel(A) represents the total amount of probability that must be allocated to A, while *Pl(A)* can be interpreted as the maximum amount of support that could be given to A.

Let m_1 and m_2 be two BBAs induced by two independent items of evidence. These pieces of evidence can be combined using *Dempster's rule of combination* which is defined as:

$$m(C) = \frac{\sum_{A \cap B = C} m_1(A) \times m_2(B)}{1 - \sum_{A \cap B = \phi} m_1(A) \times m_2(B)} \tag{3}$$

Combining BBAs using Dempster's rule of combination is possible only if the sources of belief are not totally contradictory which means that there exist two subsets $A \subseteq \Omega$ and $B \subseteq \Omega$ with $A \cap B \neq \phi$ such that $m_1(A) > 0$ and $m_2(B) > 0$.

2.1 Discounting

When an information source *S* is used in the belief function framework, its reliability can be taking into account by the *discounting operation* in which the original BBA *m* is weakened by a discounting rate $\alpha \in [0,1]$. The resulting BBA $^{\alpha}m$ is defined by Shafer [1] and Smets [8]:

$$\begin{cases} ^{\alpha}m(A) = (1-\alpha)m(A) & \forall A \in \Omega, A \neq \Omega \\ ^{\alpha}m(\Omega) = \alpha + (1-\alpha)m(\Omega) \end{cases} \tag{4}$$

The coefficient (1- α) cab be regarded as a confidence degree one has in the source of information. If α = 1, it means that *S* is not reliable and the information provided by this source is discarded. On the other hand, $\alpha = 0$ indicates that the *S* is fully reliable and the belief function remains unchanged.

2.2 Decision Making

The approach adopted in this paper for decision making is based on the pignistic transformation which is defined in the TBM. By uniformly distributing the mass of belief $\mathrm{m(A)}$ among its elements for all $A \subseteq \Omega$, a pignistic probability distribution is defined as:

$$\mathrm{BetP}(\omega) = \sum_{\{A \subseteq \Omega, \omega \in A\}} \frac{1}{|A|} \cdot \frac{\mathrm{m(A)}}{1 - \mathrm{m}(\phi)}, \quad \forall \omega \in \Omega \tag{5}$$

where $|A|$ is the cardinality of subset A and for normal BBAs, would be $m(A)$. Finally a test sample is assigned to class with the largest pignistic probability.

3 The Proposed Method

Fig. 1 shows the architecture of our proposed method. There are two main phases involved in the implementation of the method including relabeling the learning data and classifying an input test sample by combining decisions of neural networks trained on the learning data with new labels.

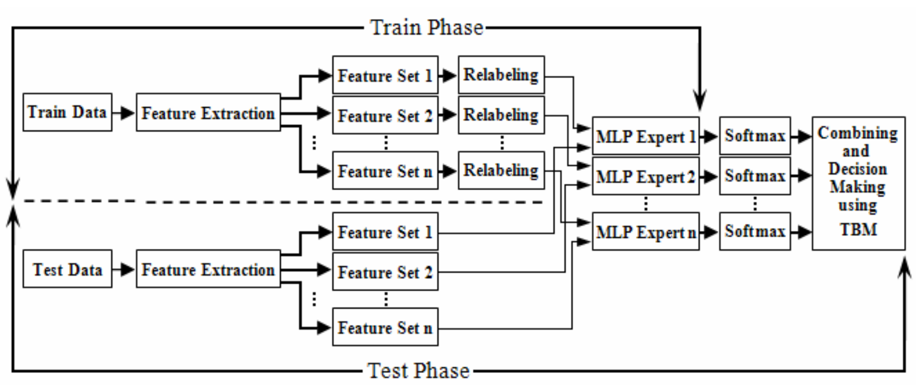


Fig. 1. Architecture of the proposed classification scheme. In the training phase, crisp and soft labels are assigned to the learning data and MLPs are trained on the data with new labels. In the test phase, the outputs of the MLPs are converted in the form of BBAs by making use of softmax operator. Final decision on a given test sample is made by combining the experts' beliefs with Dempster's rule of combination and using pignistic probability.

3.1 Relabeling

Let $\Omega = \{\omega_1,\ldots,\omega_M\}$ be a set of *M* classes and $X=[x_1,\ldots,x_n]$ be a data point described by *n* features in the training set which is associated to one class in Ω with certainty. The goals of this stage are (i) to detect inconsistencies in the labels of the training data using the supervised information carried by the data and, (ii) to reassign each train sample to just one main class or any subset of the main classes based on the level of ambiguity concerning the class membership of that sample.

Let *P* be a $M \times n$ matrix containing prototype vectors of the main classes and let $D_i = [d_{i1},\ldots,d_{iM}]$ be the set of distances between train sample X_i and the *M* prototypes according to some distance measure (e.g. the Euclidian one). Initial label of X_i is ignored and by utilizing the information provided by the vector D_i, uncertainty detection and class reassignment for this sample are performed in a three-step procedure:

Step 1: The minimum distance between X_i and the class prototypes is taken from the vector D_i,

$$d_{ij} = \min(d_{ik}), \quad k = 1,\ldots,M \tag{6}$$

and d_{ij} is called d_{min}.

Step 2: A value $0 < \mu_k \leq 1$ is calculated for each of *M* classes using the following function:

$$\mu_k(X_i) = \frac{d_{\min} + \beta}{d_{ik} + \beta}, \quad k = 1,\ldots,M \tag{7}$$

in which $0 < \beta < 1$ is a small constant value and ensures that the utilized function allocates a value greater than zero to each of *M* classes even if $d_{\min} = 0$. μ_k is a decreasing function of the difference between $d_{\min}$ and d_k and has values close to 1 for small differences.

Step 3: A threshold value $0<\tau<1$ is defined and based on the level of ambiguity regarding the class membership of the train sample X_i, this sample may be assigned to (i) a set of classes if the corresponding values of μ_k for these classes are greater than or equal to τ or (ii) just one main class, which has the closest prototype to X_i and $\mu_k = 1$, if X_i be far away from other main classes' prototypes.

Close distances between a train pattern and some of the class prototypes can be interpreted as an indication of ambiguity in the pattern's label and in such cases a soft label is assigned to that train pattern. The above procedure is repeated for all training data and for all feature spaces.

Since several representations of the data are employed to provide complementary information, it can be expected that if a soft label has been assigned to a train sample in one of the feature spaces, this sample could belong to one main class or a subset of the main classes with less uncertainty in the other feature spaces. In this way, the negative effects of crisp but imperfect labels of some learning samples on the classification performance can be reduced.

3.2 Training and Classification

MLP neural network is used as base classifier. The learning set of each feature space, which consists data with new crisp or soft labels, is employed to train a base classifier. Since different types of features are extracted from the data, each feature space has its own level of uncertainty in class labels. As a result, after the relabeling procedure, the number and type of classes in each feature space could be different from the other one and base classifiers with different models (different number of output nodes) are trained on these feature spaces. In this way, diversity among the base classifiers can be achieved.

In the test phase, same types of features as the training stage are extracted from the test data and they are applied to their corresponding trained base classifiers. The output values of each base classifier can be interpreted as measures of confidence associated with different decisions. For combining the evidences induced by the feature spaces using Dempster's rule of combination, the decision of each base classifier should be converted in the form of BBA. This can be done through normalizing the outputs of the base classifiers by a softmax operator as:

$$m_i(\{\omega_j\}) = \frac{\exp(O_{ji})}{\sum_{j=1}^{C} \exp(O_{ji})}, \qquad j = 1, \ldots, C \tag{8}$$

where O_{ji} is the jth output value of the ith base classifier, C is the number of classes in the ith feature space after the relabeling stage and $m_i(\{\omega_j\})$ is the mass of belief given to class ω_j.

Note that, although our method allows to compute a BBA m_i that could has any focal set over the set of the main classes, the BBA contains only a set of states which have corresponding classes in the *i*th feature space after the relabeling stage.

Before combining the opinions of different evidences, the belief function of each classifier is weakened by its discounting rate (the method of evaluation of the discounting factor will be explained in Section 3.2.1) and the resulting BBAs are merged using Dempster's rule of combination. The decision about the class of a given test sample is then made using the pignistic probability derived from the final BBA by the pignistic transformation.

Note that, although the main contribution of our method is to classify data with imperfect labels, its application can be extended to classification problems which involve heavily overlapping class distributions and nonlinear class boundaries.

3.2.1 Evaluation of the Discounting Factor

To assess the reliability of information sources provided by different feature spaces, the method proposed by Eloudi et.al. [9] is used. This method is based on the TBM and for finding the discounting factor, minimizes the distance between the pignistic probabilities computed from the discounted beliefs and the actual values of data. The outline of this approach is summarized below.

Let $\Omega=\{\omega_1,...,\omega_M\}$ be a set of classes and $\chi=\{X_1,...,X_n\}$ be a set of *n* samples. Let BBA $m(X_j)$ be the normalized belief function for sample X_j which is defined on the set of classes. The class of each sample is known and $c_j \in \Omega$ denotes the class of sample X_j. Let $BetP^{\alpha}(X_j)$ denotes the pignistic probability obtained from the discounted BBA ${}^{\alpha}m(X_j)$. For each sample X_j, an indicator function $\delta_{j,i}$ is defined as:

$$\delta_{j,i}=\begin{cases}1 & \text{if } c_j=\omega_i,\\ 0 & \text{otherwise,}\end{cases} \tag{9}$$

and sum of the Euclidian distances between the pignistic probabilities computed from the discounted BBAs and the indicator functions, is calculated as follows:

$$\text{TotalDist}=\sum_{j=1}^{n}\sum_{i=1}^{M}(BetP^{\alpha}(X_j)(\omega_i)-\delta_{j,i})^2 \tag{10}$$

The discounting factor α that minimizes the above distance is given by:

$$\alpha=\min\left(1,\max\left(0,\frac{\sum_{j=1}^{n}\sum_{i=1}^{M}(\delta_{j,i}-BetP(X_j)(\omega_i))BetP(X_j)(\omega_i)}{n/M-\sum_{j=1}^{n}\sum_{i=1}^{M}(BetP(X_j)(\omega_i))^2}\right)\right) \tag{11}$$

4 Experimental Results

In this section, we report experimental results on artificial and real datasets to highlight the main aspects of our proposed method. In our experiments, the centers of the main classes are taken by averaging the learning data of each class and the resulting vectors are considered as the main classes' prototypes. A value of $\beta = 0.01$ was adopted in the relabeling stage and validation sets were used to calculate the discounting factor.

4.1 Artificial Data

We used a two-dimensional data so that the results could be easily represented and interpreted. The dataset was made of three classes with equal sample size, Gaussian distribution and common identity covariance matrix and 150, 300 and 1500 samples were generated independently for training, validation and testing sets, respectively. The center of each class was located at one of the vertices of an equilateral triangle. In order to use different representations of the data, the classes were transferred to their near vertices in clockwise direction and in this fashion two other feature spaces were generated. In each feature space, a unique subset of each class overlapped with data of other classes. It means that the high uncertainty pertaining to class membership of a pattern in one of the feature spaces can be reduced because this pattern may be located in non-overlapping or less ambiguous areas in the other feature spaces. To evaluate the performance of the proposed method in classification tasks with different levels of uncertainty in class labels, the length of the equilateral triangle (the distance between the neighborhood classes) was varied in {1,2,3} and in this way, three cases from strongly overlapping to almost separated classes were studied. Fig. 2 represents graphically the explained above procedure for generating the artificial data.

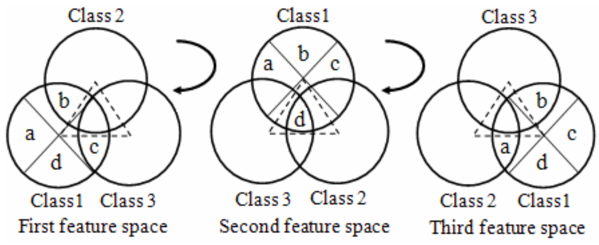


Fig. 2. Generic representation of the artificial data. For generating new feature spaces, the classes are transferred to their near corners in the clockwise direction. To demonstrate how different parts of a class overlap with other classes in different feature spaces, class 1 is divided to four parts and the positions of these parts in the three feature spaces are shown.

4.1.1 Soft and Crisp Label Generation

There are two main issues in the relabeling stage including choice of the main classes' prototypes and selecting the threshold value (τ). Although an appropriate selection of prototype can play an important role in accurate assignments of crisp or soft labels to data, we focus our attention on the influence of considering uncertainty in the labels of data on the performance of a classification method that confronts data with imperfect class labels or highly overlapping class distributions.

In order to examine how training samples with different levels of uncertainty in their labels are treated in the relabeling procedure, the results of partitioning the first training set with $\tau = 0.8$ is demonstrated in Fig. 3. Each partition is represented by a convex hull and its class label indicates that the samples of that partition were assigned to which subset of the main classes. It can be seen that soft labels were assigned to samples situated in the boundaries of the main classes or those located at ambiguous regions.

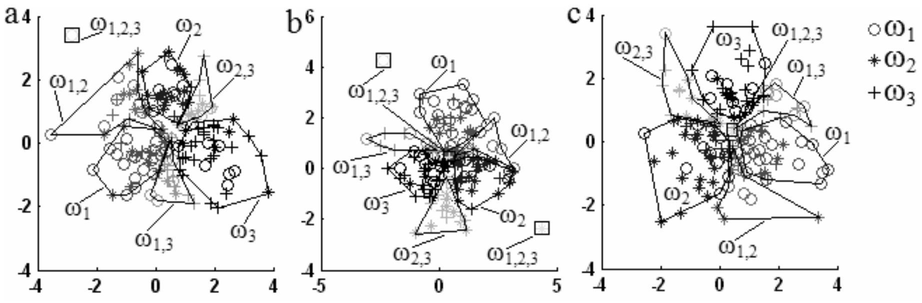


Fig. 3. Representation of partitioning the first learning set into crisp and soft subsets by the proposed relabeling approach with $\tau = 0.8$. a) first feature space, b) second feature space, c) third feature space

4.1.2 Performance Comparison

The performance of our proposed method was compared to three MLP neural networks trained separately on one of the feature spaces and ensemble networks constructed by merging the decisions of the single MLPs using three fixed combining methods (Averaging, Product and Max rules). The single MLPs and the ensemble networks discarded the possible uncertainties in class labels and employed data with initial certain labels. The MLPs were trained by Levenberg-Marquardt algorithm with default parameters and 80 epochs of training and had one hidden layer. Each neural network was trained 50 times with random initializations. To evaluate the performance of our proposed method based on different threshold values, we generated 19 training sets from each original training data by varying τ from 0.05 to 0.95 with a step size of 0.05.

The average test error rates of the employed classifiers on the three datasets and for different number of hidden nodes are represented in Fig. 4. Note that, the classification results of our proposed method with the best τ for each dataset are shown. As can be seen, our method yields considerably better classification results than the other classifiers when the first dataset, as the most difficult set with highest level of ambiguity in class labels, was used (Fig. 4 (a)). As shown in Fig. 4(b) and Fig. 4(c), the differences between the test performances of our proposed scheme and the three ensemble networks on the second and third datasets are small. So, it can be concluded that there is not much benefit to be gained from considering uncertainty in perfectly labeled data or in a classification problem with separated classes.

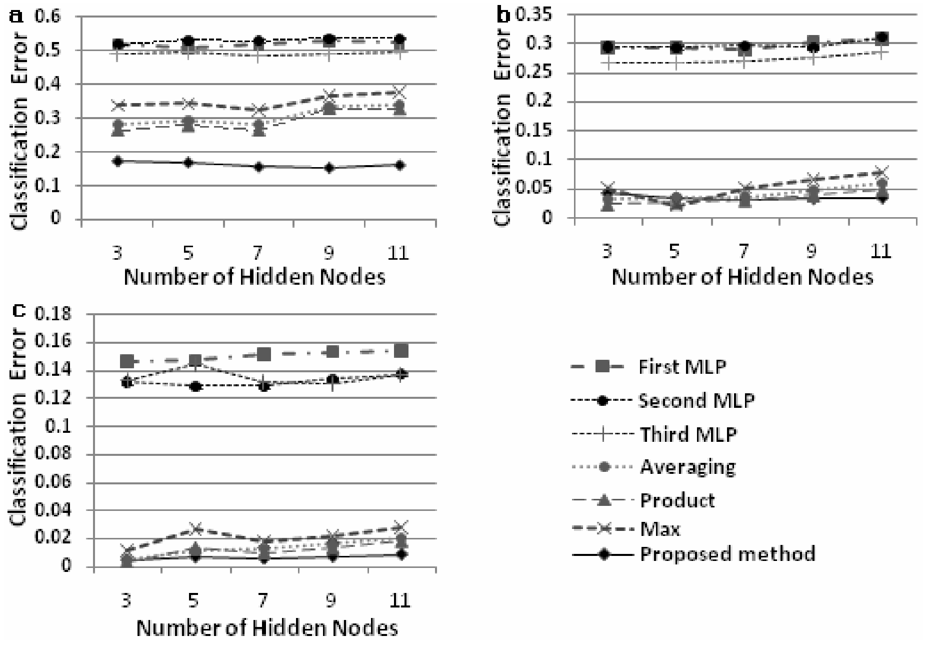


Fig. 4. Classification error rates as a function of number of hidden nodes for single MLPs, ensemble networks constructed by combining the MLPs using fixed combining method and our proposed method. a) first training set (τ=0.8 for our method), b) second training set (τ=0.75 for our method), c) third feature space (τ=0.7 for our method).

4.2 Real Data

We applied our proposed method to the problem of classifying circular knitted fabric defects. The data consisted of five classes of knitted defect samples and defect-free fabric images (Fig. 5).

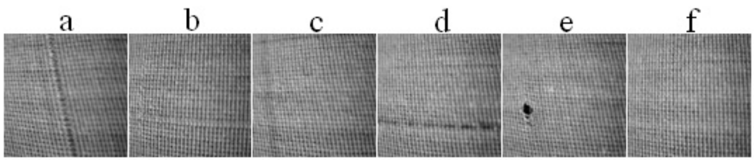


Fig. 5. Samples of fabric images in our six-class problem. a) Vertical Strip, b) Horizontal Strip, c) Soil freckle, d) Crack, e) Hole, f) Defect-free.

As can be seen from Fig. 5, four classes belong to either horizontal or vertical defects which means that the extracted features from samples of these classes can make overlapping areas in feature space. Moreover, little differences between samples of some defect classes and defect-free images may also cause vagueness in information provided by the learning data.

The feature vectors were computed from 128×128 pixels gray level images in three stages including wavelet decomposition, binary thresholding and morphological processing. Three wavelet filters of Daubechies family (db2, db5, db10) and three levels of decomposition were used based on extensive experiments. Horizontal and vertical detailed subimages at level 3 were used for further analysis since most of the defective fabrics had either horizontal or vertical defects. Detailed subimages were then converted to binary, binary level can vary from 0 to 1 and it was chosen to be 0.2 in our work, and a morphological filter (*Opening*) was applied to them to eliminate irrelevant parts which could be considered as defective areas. White areas in the final subimages were considered as defects and total number of white pixels in the final horizontal and vertical images made the first and second dimensions of the feature vector, respectively. Fig.6 illustrates three two-dimensional feature spaces which were achieved using the above feature extraction approach.

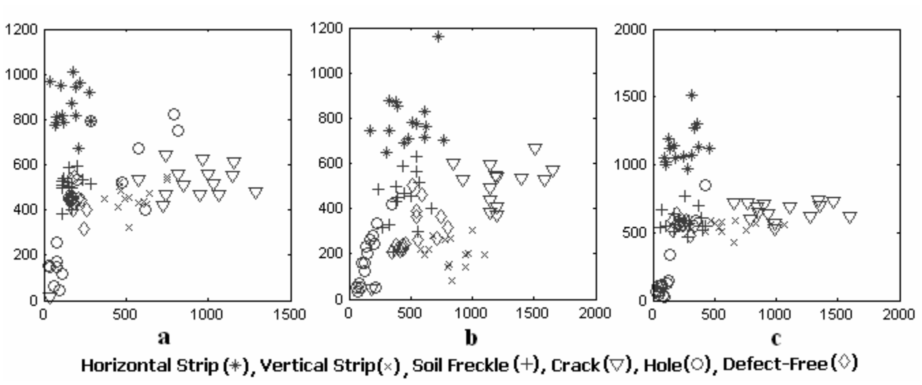


Fig. 6. Representations of the fabric samples in the two-dimensional feature spaces obtained by making use of three wavelet filters. a) db2, b) db5, c) db10.

The dataset involved 90 samples and all classes had equal number of patterns. Overall classification performance was evaluated using a K-fold cross validation method with K = 5. The data was divided into five subsets and in each round, three of the five subsets were employed for training while the fourth and fifth were used for validation and testing, respectively. The above procedure was repeated for all five subsets and the average classification rate on the test patterns was considered as a figure of merit. Similar to experiments carried out using the artificial dataset, we compared the performance of our proposed method on the real data with those of single and ensemble neural networks that relied on the initial imperfect labels. All MLPs were trained 100 times with random initial weights and 50 epochs of training and their architectures were similar to the former experiment.

Table 1 gives the average test error rates of the proposed method (with the best threshold value (τ=0.75)), the ensemble networks and the single MLPs as a function of number of hidden nodes.

Table 1. Error rates as a function of number of hidden nodes for single MLPs, ensemble networks and our method ($\tau = 0.75$). The minimum error rate of each network is typed in bold.

No. of hidden neurons	3	5	7	9	11
Average error rate (%)					
First MLP	47.1	36.72	33.79	**32.37**	32.39
Second MLP	41.42	30.58	**27.63**	28.34	29.15
Third MLP	41.19	34.87	33.81	**33.75**	34.09
Averaging	28.11	20.34	**18.25**	18.33	18.25
Product	36.08	19.34	**17.98**	18.21	18.95
Max	32.69	24.45	22.78	**22.58**	22.83
Proposed method	26.29	18.68	17.6	**17.13**	17.39

Small differences between error rates of the proposed method and the ensemble networks can be interpreted by considering the poor choice of the main classes' prototypes (center of classes) in the relabeling stage of our method. As mentioned earlier, our major interest is to investigate the effect of considering uncertainty in class labels in improving the classification results and utilizing complementariness between information sources in the framework of the D-S theory. So, the expectation is that by employing more complex algorithm for generating prototypes of the main classes, which is an open area of research, better classification results would be achieved using the proposed method

We used the real data to examine the effect of utilizing discounting strategy on the performance of the proposed method and to explore how diverse classifiers were generated from the feature spaces using our proposed relabeling approach. Error reduction rates obtained by employing the discounting factor in our method for different threshold values and the best number of hidden neurons are presented in Table 2. Overall, discounting brought better performances compared with the situation that all information sources assumed to be fully reliable.

Table 2. Error reduction rates (%) achieved by employing the discounting factor in the proposed method as a function of the threshold value for the best number of hidden nodes (9 hidden nodes)

Threshold value (τ)	0.15	0.35	0.55	0.75	0.95
Error reduction rate (%)	0.16	0.39	4.94	3.87	0.22

Table 3 lists the generated classes from each feature space after the relabeling stage for the best threshold value (τ=0.75). Here, we randomly selected one of the training sets produced by the five-fold cross validation. It can be seen that the number and type of generated soft class labels for each training set are almost different from another one which indicates that the ensemble network was made by a set of diverse classifiers. Apart from the influence of τ on the label generation procedure, form of the produced class labels is related to the number of overlapping areas and amount of uncertainty pertaining to samples of each area.

Table 3. Generated classes from the three feature space after the relabeling stage for the best threshold value (τ =0.75)

Produced classes										
Feature Set 1	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	$\omega_{2,4}$	$\omega_{3,6}$	$\omega_{1,3,6}$	$\omega_{3,5,6}$	-
Feature Set 2	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	$\omega_{2,6}$	$\omega_{3,5}$	$\omega_{3,6}$	-	-
Feature Set 3	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	$\omega_{2,4}$	$\omega_{2,6}$	$\omega_{3,6}$	$\omega_{2,3,6}$	$\omega_{1,2,3,6}$

5 Conclusion

In this paper, a method for handling imperfect labels using belief functions has been presented. By extracting different types of features from data, the proposed method takes advantage of information redundancy and complementariness between sources. In each feature space and by making use of the proposed relabeling technique, the initial labels of the learning data are ignored and each train pattern is then reassigned to a class with crisp or soft label based on its closeness to prototypes of the main classes. MLP neural network is used as base classifier and its outputs are interpreted as BBA and in this way, partial knowledge about the class of a test pattern is encoded. The BBAs are then discounted based on the reliability of the base classifiers in identifying validation patterns and are pooled using Dempster's rule of combination.

Experiments carried out on controlled simulated data and a dataset of knitted fabric defects. It was shown that by considering the ambiguity in labels of the data, our method can outperform classifiers that rely on the initial imperfect labels.

References

1. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press, Princeton (1976)
2. Smets, P., Kennes, R.: The Transferable Belief Model. Artif. Intell. 66, 191–243 (1994)
3. Rogova, G.: Combining the result of several neural network classifiers. Neural Networks 7(5), 777–781 (1994)
4. Denoeux, T.: A k-Nearest Neighbor Classification Rule Based on Dempster-Shafer Theory. IEEE Trans. Syst., Man, Cybern. 25(3), 804–813 (1995)
5. Denoeux, T.: A Neural Network Classifier Based on Dempster-Shafer Theory. IEEE Trans. Syst., Man, Cybern. A, Syst., Humans 30, 131–150 (2000)
6. Basir, O., Karray, F., Zhu, H.: Connectionist-Based Dempster-Shafer Evidential Reasoning for Data Fusion. IEEE Trans. on Neural Net. 16, 1513–1529 (2005)
7. Quost, B., Denoeux, T., Masson, M.-H.: Pairwise classifier combination using belief functions. Pattern Recognition Letters 28, 644–653 (2007)
8. Smets, P.: Belif Functions: The Disjunctive Rule of Combination and the Generalized Bayesian Theorem. International Journal of Approximate Reasoning 9, 1–35 (1993)
9. Elouedi, Z., Mellouli, K., Smets, P.: Assessing Sensor Reliability for Multisensor Data Fusion Within the Transferable Belief Model. IEEE Trans. Syst., Man, Cybern. B 34, 782–787 (2004)

Stability and Topology in Reservoir Computing

Larry Manevitz and Hananel Hazan

Department of Computer Science, University of Haifa, Mount Carmel, Haifa 31905, Israel
[manevitz,hhazan01]@cs.haifa.ac.il

Abstract. Recently Jaeger and others have put forth the paradigm of "reservoir computing" as a way of computing with highly recurrent neural networks. This reservoir is a collection of neurons randomly connected with each other of fixed weights. Amongst other things, it has been shown to be effective in temporal pattern recognition; and has been held as a model appropriate to explain how certain aspects of the brain work. (Particularly in its guise as "liquid state machine", due to Maass et al.) In this work we show that although it is known that this model does have generalizability properties and thus is robust to errors in input, it is NOT resistant to errors in the model itself. Thus small malfunctions or distortions make previous training ineffective. Thus this model as currently presented cannot be thought of as appropriate as a biological model; and it also suggests limitations on the applicability in the pattern recognition sphere. However, we show that, with the enforcement of topological constraints on the reservoir, in particular that of *small world* topology, the model is indeed fault tolerant. Thus this implies that "natural" computational systems must have specific topologies and the uniform random connectivity is not appropriate.

Keywords: Reservoir Computing, Small world topology, robustness, Machine Learning.

1 Introduction

Recently Jaeger (Jaeger, "The ëcho state" approach to analysing and training recurrent neural networks, 2001), Maass (Maass, Natschläger, & Markram, 2002) and others have put forth the paradigm of "reservoir computing" as a way of computing with highly recurrent neural networks. This reservoir is a collection of neurons randomly connected with each other of fixed weights. Amongst other things, it has been shown to be effective in temporal pattern recognition; and has been held as a model appropriate to explain how certain aspects of the brain work. (Particularly in its guise as "liquid state machine", due to Maass et al.)

This is particularly impressive as processing in artificial neurons typically is a-temporal. This is because the underlying basic neuronal model, that of McCullough-Pitts (McCullough & Pitts, 1943) is atemporal by nature. As a result, most applications of artificial neural networks are related in one way or another to static pattern recognition. On the other hand, it has long been recognized in the brain science community that the McCullough-Pitts paradigm is inadequate. Various models of differing complexity have been promulgated to explain the temporal capabilities (amongst other things) of natural neurons and neuronal networks.

However, during the last decade, computational scientists have begun to pay attention to this issue both from the neurocomputational and biological perspectives e.g. (Maass W. , 1999; Maass, Natschläger, & Markram, 2002; Maass, Natschlger, & Markram, 2004; Fernando & Sojakka, 2005; Jaeger, The "echo state" approach to analysing and training recurrent neural networks, 2001), and investigations as to the computational capabilities of various models are being investigated.

Two such models, the “echo state machine” (Jaeger, The "echo state" approach to analysing and training recurrent neural networks, 2001) and the “Liquid State Machine” (see *Fig. 1.*) (Maass, Natschlger, & Markram, 2004; Maass, Natschläger, & Markram, 2002), have had substantial successes recently. These two models are identical on the abstract level and have recently been renamed “reservoir computing” (Lukosevicius & Jaeger, 2009). In these models there is a somewhat different paradigm of computation. Information is stored, not in "attractors" as is usually assumed in recurrent neural networks, but in the reverberating activity pattern in a sufficiently recurrent and inter-connected network. This information can then be retrieved by any sufficiently strong classifying detector. The idea is that the history of, e.g. timings of rocks thrown into a pond of water, is completely contained in the wave structure.) Moreover, the "persistence of the trace" (or as Maass put it, the "fading memory") allows one to recognize at a temporal distance the signal that was sent to the liquid; and sequence and timing affects of inputs.

This is an exciting idea; and, e.g. Jaeger, Maass and his colleagues have published a series of papers on it. Amongst other things, they have recently shown that once a detector has been sufficiently trained at any time frame, it is resilient to noise in the input data; and so it can be used successfully for generalization. (Maass, Natschläger, & Markram, 2002; Fernando & Sojakka, 2005). In particular, experiments have been performed for speech recognition.

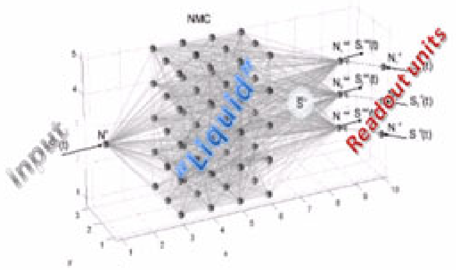


Fig. 1. Liquid State Machine (*figure* taken from (Maass, Natschläger, & Markram, 2002))

However, there is a claim that this abstraction is faithful to the potential capabilities of the natural neurons and thus is explanatory to some extent from the viewpoint of computational brain science. It is this issue we address in this paper. Note that one of the underlying assumptions is that the detector works without memory; that is the detector should be able to classify based on instantaneous static information; i.e. by sampling the liquid at a specific time. That this is theoretically possible is the result of looking at the dynamical system of the liquid and noting that it is sufficient to cause the divergence of the two classes in the space of activation.

Note that the detector systems (e.g. a back propagation neural network, a perceptron or an SVM) are not required to have any biological plausibility; either in their design or in their training mechanism, since the model does not try to account for the way the information is *used* in nature.

Despite this, since natural neurons exist in a biological and hence noisy environment, for these models to be successful in this domain, they must be robust to various kinds of noise. As mentioned above (Lukosevicius & Jaeger, 2009) (Maass, Natschläger, & Markram, 2002) addressed one dimension of this problem by showing that the systems are in fact robust to noise in the input. Thus small random shifts in a temporal input pattern will not affect these models to recognize the pattern. From a machine learning perspective, this means that the model is capable of generalization.

However, there is another component to robustness; that of the components of the system itself.

In this paper we report on experiments performed with various kinds of "damage" to these models and unfortunately have shown that, e.g. the LSM with any of the above detectors is *not* resistant, in the sense that small damages to the LSM neurons reduce the trained classifiers dramatically, even to essentially random values.

Seeking to correct this problem, we experimented with different architectures of the liquid. The essential need is that there should be sufficient recurrent connections so that on the one hand, the network maintains the information in a signal, while on the other hand it separates different signals. The models typically used are random connections; or those random with a bias towards "nearby" connections. Our experiments with these topologies show that the network is very sensitive to damage because the recurrent nature of the system causes substantial feedback.

Taking this as a clue, we tried networks with "hub" or "small world" (Albert-László & Réka, 1999; Albert-László & Réka, 1999; Bianconi G, 2001; Albert R, 2000) architecture. This architecture has been claimed (Danielle & Bullmore, 2006; Chklovskii, 2009) to be "biologically feasible".

The intuition was that the hub topology, on the one hand, integrates information from many locations and so is resilient to damage in some of them; and on the other hand, since such hubs follow a power rule distribution, they are rare enough that damage usually does not affect them directly. This intuition was in fact borne out by our experiments.

2 Non Robustness of the Models

2.1 The Experiments

To test this resistance to noise, we downloaded the code of Maass et al from his laboratory site[1] and then implemented two kinds of damage to the liquid. We also reimplemented the LSM code so that we could handle variants. These models use a kind of basic neuron that is of the "leaky integrate and fire" (LIF) variety and in Maass'

[1] A neural *C*ircuit *SIM*ulator: http://www.lsm.tugraz.at/csim/

work, the neurons are connected randomly. In addition, some biologically inspired parameters are added: 20% inhibitory and a connectivity constraint giving a preference to geometrically nearby neurons over more remote ones. (For precise details on these parameters, see: neural *C*ircuit *SIM*ulator[1]) External stimuli to the network were always sent to 15% of the neurons, always chosen to be excitatory neurons. Initially, we experimented with two parameters: (i) the percentage of neurons damaged (ii) the kinds of damages.

The kinds were either transforming a neuron into a "dead" neuron; i.e. one that never fires or transforming a neuron into a "generator" neuron ,i.e. one that fire as often as its refractory period allows it, regardless of its input.

We did experiments with different kinds of detectors: Adaline (Widrow & Hoff, 1960), Back-Propagation, SVM and Tempotron (Gütig & Sompolinsky, 2006).

Classification of new data could then be done at any of the signal points; We ran experiments as follows: we randomly chose twenty temporal inputs; i.e. random sequences of 0s and 1s of length 45, corresponding to spike inputs over a period of time; and trained an LSM composed of 240 integrate and fire neurons like in (Maass, Natschläger, & Markram, 2002) to recognize ten of these inputs and reject the other ten, each choice of architecture was run 666 times varying the precise connections randomly.

We tested the robustness of the recognition ability of the network with the following parameters:

- The neurons in the network were either leaky integrate and fire neurons (Maass W. , 1999) or Izhikevich (Izhikevich, 2003) style neurons.
- The average connectivity of the networks was maintained at about 20% chosen randomly in all cases although with different distributions.
- The damages were either "generators," i.e. the neurons issued a spike whenever their refractory period allowed it; or they were "dead" neurons that could not spike.
- The degree of damage was systematically checked at 1%, 2%...15% in randomly chosen neurons.

The results that shown in tables throughout the paper are in percents, over the (666) repeated tests. 100% indicates that all the 20 vectors of one test, over 666 repetitions of the test were fully recognize correctly. 50% indicates that only half the vectors over 666 times were recognized (corresponding to chance baseline).

2.2 Results

First, there was not much difference between the detectors; so eventually we restricted ourselves to the Back-Propagation detector which had data points of 30 randomly sampled time points of the entire liquid. (To be fair, none of units of the liquid input were accessed by the detectors allowed to be input neurons of the liquid.)

It turned out that while the detector was able to learn the randomly chosen test classes successfully with sufficient average connectivity almost any kind of damage caused the detector to have a very substantial decay in its detecting ability (See *Table* 1.).

Table 1. Maass's implementation: distribution preferring local connections

Damage	Non	5%	10%
Dead Neurons	90.45%	80.35%	60.3 %
Noisy Neurons	92.01%	59.08%	53.8%

Reimplementing the LSM to allow for different connectivities; showed the same basic responses. In fact, when the network is connected randomly without bias for geometric closeness, the network is even more sensitive (Compare *Table* 1. and *Table* 2.). After our later experiments, we returned to this point (see concluding remarks, below).

Table 2. 10% random connections[2]

Damage	Non	1%	5%	10%
Dead Neurons	100%	53%	53%	50%
Noisy Neurons	100%	55%	53%	52%

In *Fig. 2.* we illustrate the difference in reaction of the network by a raster (ISI) display. Note that with 10% damage, it is quite evident to the eye that the network diverges dramatically from the noise free situation. In *Table* 3. one can see this as well with 5% noise for purely random connectivity. Actually, with low degrees of damage the detectors under even the Maass connectivity show dramatic decay in recognition although not to the extremes of random connectivity. (See *Table* 2.) These results were robust and repeatable under many trials and variants.

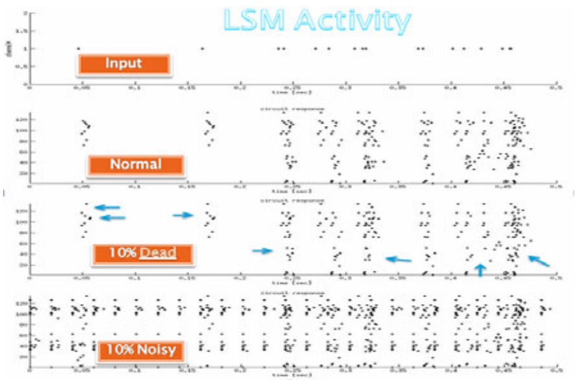


Fig. 2. Maass LSM (a) normal operation (b) with 10% dead damage (c) with 10% noise

Accordingly, we conclude that the LSM, either as purely defined with random connectivity, or, as implemented in (Maass, Natschlger, & Markram, 2004) cannot serve as a biologically relevant model.

[2] For all the Tables that shown in this paper, 50% is the baseline of random classification.

3 Topological Modifications and Restoration of Robustness

3.1 Different Kinds of Basic Neurons

In attempts to restore the robustness to damage, we experimented with the possibility that a different kind of basic neuron might result in a more resilient network. Accordingly, we implemented the LSM with various variants of "leaky integrate and fire neurons" e.g. with history dependent refractory period (Manevitz & Marom, 2002) and by using the model of neurons due to Izhikevich (Izhikevich, 2003). The results under these variants were qualitatively the same as the standard neuron. (The Izhikevich model produces a much more dense activity in the network and thus the detector was harder to train but in the end the network was trainable and the results under damage were very similar.).

3.2 Allowing Detectors to Have Memory

In trying to consider how to make the model more robust to damage, we focused first on the fact that the detector has no memory. Perhaps, if we allow the detector to follow the development of the network for some time amount, both in training and running, it would be more robust. To check this, we took the most extreme other case; we assumed that the detector system in fact takes as input a full time course of its input neurons for about 30 iterations. This means that instead of a NN with input of 204; we had one with 30 times 204 time course inputs. It seemed reasonable that (i) with so much information, it should be relatively easy to train the detector (ii) one could hope that damage in the liquid would be local enough that over the time period, the detector could correct for it. In order to test this, we re-implemented the LSM to allow for the time entry.

Our detector was trained and tested as follows. There were 204 output units. At a "signal point" each of them was sampled for the next 30 iterations and all of these values were used as a single data point to the detector. Thus the detector had 204 times 30 inputs. We chose separate detector points typically at intervals of 80. We then used back propagation on these data points. This means that eventually the detector could recognize the signal at any of the "signal points" after training there was no particular importance to the choice of separation of the signal points except that there was no overlap between the data points. While we did not control for any connection between the intervals of data points (i.e. 80, and we also checked other time intervals) and possible natural oscillations in the network, we do not believe there was any. As anticipated, there was no significant trouble in training the network to even 100% of recognition of the training data.

The "detectors" were three level neural networks, trained by back-propagation. We also did some experiments with the Tempotron (Gütig & Sompolinsky, 2006); and with a simple Adaline detector (Widrow & Hoff, 1960). Training for classification could be performed in the damage-less environment successfully with any of these detectors.

We exhaustively ran tests on these possibilities. Some sample results with 5% and 10% damage for the neural network detectors are presented in the *Fig. 4.* Through *Fig. 9.* below. (Since the results for the other detectors were similar, we did not run as many tests on them)

Table 3. 5% random connectivity

Damage	Non	1%	5%	10%
Dead Neurons	100%	61%	58%	56%
Noisy Neurons	100%	60%	58%	57%

In all of these tests, following Maass, we assumed that approximately 20% of the neurons of the liquid were of the inhibitory type. The architecture of the neural network detector was 204 input neurons (which were never taken from the neurons in the LSM which were also used as inputs to the LSM.) 100 hidden level neurons and one neuron for the output. Results running the Maass et al. architecture are presented in Table 1. and can be compared with a random connected network of 20% average connectivity. See Table 4.

Table 4. 20% random connectivity

Damage	Non	1%	5%	10%
Dead Neurons	100%	79%	49%	49%
Noisy Neurons	100%	97%	71%	63%

The bottom line was that even with low amounts damage and under most kinds of connectivity, the networks would fail; i.e. the trained but damaged network loss of function was very substantial and in many cases could not perform substantially differently from a random selection.

3.3 Changing the Architecture

Our next approach, and ultimately the successful one, was to experiment with different architectures. We looked at many variants of the following ideas:

1. Random Connectivity as a baseline. (Note: this is not a "straw dog". This is actually the basic definition of the LSM.)
2. Varying the amount of connectivity. Lowering the average degree of connectivity shows decreased sensitivity in all architectures. Unfortunately, lowering the connectivity also decreases the strength the network has in representability and, importantly, in the persistence of the signal. (That is, a low degree of connectivity causes the activity to die down quickly because of the lack of feedback. Thus the network is bounded in time and cannot recognize an "older" input signal.) Thus we see, as is to be expected from the analysis in (Jaeger, The "echo state" approach to analysing and training recurrent neural networks, 2001; Maass, Natschlger, & Markram, 2004)that a higher connectivity gives a larger set of "filters" that separate signals, but on the other hand makes it more sensitive to changes. In any case, even with low connectivities, the random topology was not robust; nor was the Maass topology. (While not at random levels of identification, as we have seen, e.g. in *Table* 1. it suffered very substantial decays with even small amounts of damages. In addition, our experiments with connectivities below 15% - 20%, show that the networks do not maintain the trace for very long. (Not shown here.)

3. "Hub" topologies (see *Table* 5.). Here we designed by hand topologies with essentially one hub. In this case, the robustness was substantially increased but the persistence was weak; and under the algorithm chosen, there were substantial disconnected components in the liquid.
4. Small world topologies (see *Table 6.*). In this system the connectivity follows a power rule law. We constructed these networks in different ways. In all cases, the number of connections was chosen based on the average connectivity desired.
5. Assign a link from a uniformly randomly chosen neuron to a second neuron chosen randomly according to a power law. In this case the input connectivity follows a power law; while the output connectivity follows a Gaussian distribution.
6. Reversing the above. In this case the input connectivity is Gaussian while the output connectivity is power law.
7. We also replaced Gaussian with uniform in the above.
8. We also tried choosing a symmetric network with Power law connectivity (ipso facto for both input and output). (Note that in this case, the *same neurons* served as "hubs" both for input and output.)
9. Finally, we designed an algorithm to allow distinct input and output connectivity but both obeying the same power law. (See algorithm1 and algorithm 2 below).

Algorithm 1. Generate a random number between min and max value with Power law distribution

```
Input: min,max, size
How_many_numbers
counterArry = array
Magnify = 5
for  i = 1  to How_many_numbers
 index = random(array.start ,
array.end)
 end_array = array.end
 candidate = array[index]
 AddCells(array , Magnify);
 for t = 0 to Magnify
  array[end_array+t]=candidate
 end for
 shuffle(array)
 output_Array[i] = candidate
 counterArry[candidate]++
end for
shuffle(counterArry)
Output output_Array,counterArry
```

Algorithm 2. Create the connectivity matrix for the liquid network using the algorithm 1

```
Input weight_Matrix
use algorithm 1 to creart
(arraylist, counterArry)
counter = 0
for i=1 to
counterArry.lenght
 for t=1 to counterArry[i]
  weight_Matrix[i,
arraylist[counter]]=true
  counter++
 end for
end for
```

One problem with the various algorithms for designing power law connectivity is that under a "fair" sampling, the network might not be connected. This means that such a network actually has a lower, effective connectivity. Since we already knew that lower connectivity results in less sensitivity to noise, we decided to eliminate this problem by randomly connecting the disconnected components (either from an input or output perspective) to another neuron chosen randomly but proportionally to the connectivity. (This does not guarantee connectivity of the graph, but makes it unlikely, so that the effective connectivity is not substantially affected.)

Table 5. One hub network

Damage	Non	1%	5%	10%
Dead Neurons	100%	93%	66.73%	64.09%
Noisy Neurons	100%	98%	84.33%	70.77%

Table 6. Power-law distribution with small worlds

Damage	Non	1%	5%	10%
Dead Neurons	100%	87%	71%	68%
Noisy Neurons	100%	88%	79%	71%

3.4 Results

All architectures with a power law distribution, whether on the input, output or both sides resulted in substantial improvements in the resistance to noise, except for case (8) where the input and output hubs were the same. This was even worse than the random baseline choice at the same connectivity. The best result was obtained in case (9) when both input and output connectivity were power law; but distinctly chosen. *Fig. 3* shows the connectivity distribution in this case. *Table 6.* shows the results in this case for some sample damages in the 20% average connectivity situation.

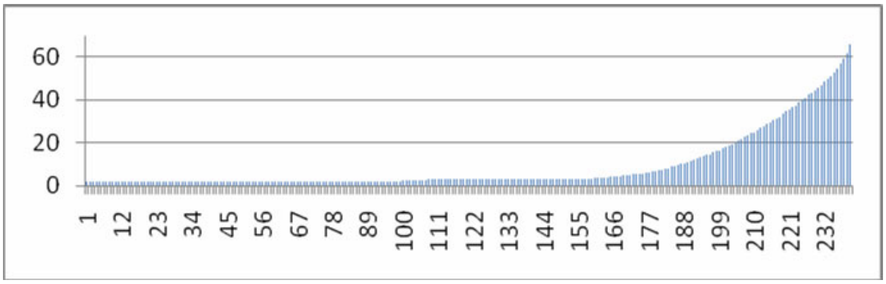


Fig. 3. Connection distribution according to the power-law

The results presented are the average of many experiments. However, since this work is about robustness, we thought it important to consider the distribution of such results over many experiments. Thus, instead of giving less revealing statistics, we display the complete histograms for the different kinds of networks under different amounts of damages. Note that for all figures and tables 50% is the random baseline.

In *Fig. 4.* through *Fig. 9.*, we display the histograms of hundreds of networks at different levels of success under each of the architectures. The horizontal axis is the accuracy of classification and the vertical axis is the histogram count.

Table 4., *Fig. 6.* and *Fig. 7.* show the distribution of damage for a random connectivity network with average connectivity of 20%.

Table 5., *Fig. 8.* and *Fig. 9.* show the distribution of damage for one hub network with average connectivity of 20%.

Table 6., *Fig. 4.* and *Fig. 5.* show the robustness results for the power law small word distribution.

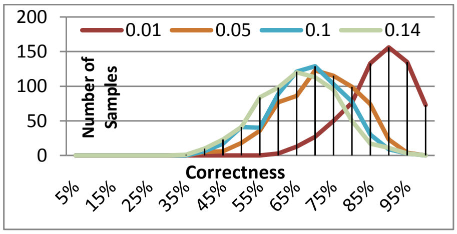


Fig. 4. Histograms of correctness results in liquid networks with different amounts of "dead" neuron damage for liquid networks with an average connectivity of 20% with a power law distribution of connectivity

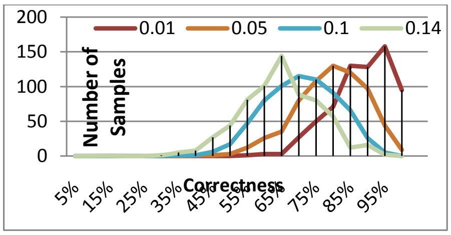


Fig. 5. Histograms of correctness results in liquid networks with different amounts of "noise generator" neuron damage for liquid networks with an average connectivity of 20%, with connectivity of a power law distribution

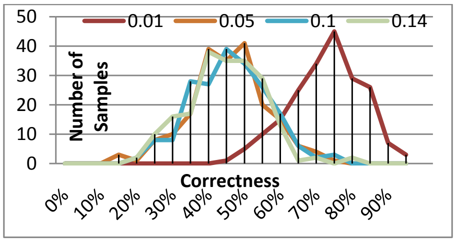


Fig. 6. Histograms of correctness results in liquid networks with different amounts of "dead" neuron damage for liquid networks with an average connectivity of 20% with a connectivity of random connections distribution

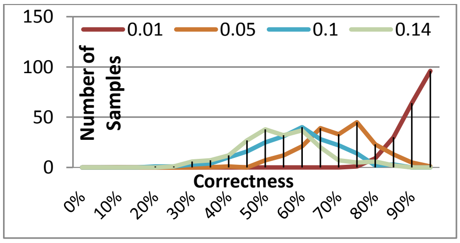


Fig. 7. Histograms of correctness results in liquid networks with different amounts of "noise generator" neuron damage for liquid networks with an average connectivity of 20% with a random connections[3] distribution of connectivity

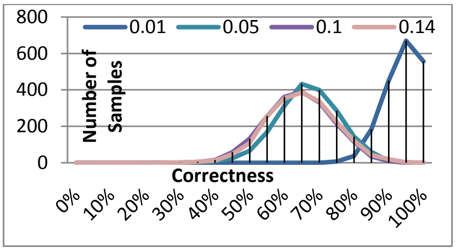


Fig. 8. Histograms of correctness results in liquid networks with different amounts of "noise generator" neuron damage for liquid networks with an average connectivity of 20%, with distribution of one hub

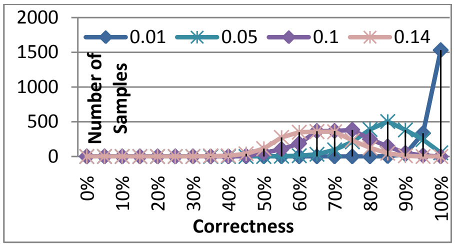


Fig. 9. Histograms of correctness results in liquid networks with different amounts of "dead" neuron damage for liquid networks with an average connectivity of 20%, with distribution of one hub

[3] For all the *Table*s that shown in this paper, 50% is the baseline of random classification.

4 Discussion

We have shown experimentally that the basic LSM is not robust to "damages" in its underlying neurons and thus without elaboration cannot be seen to be a good fit for a model for biological computation. We mention (data not shown here) that this result holds even if training is continued while the network is suffering damage. However, choosing different distributions of the connectivity can result in more robust maintenance of the pertinent information over time.

In the papers (Danielle & Bullmore, 2006; Chklovskii, 2009), a distribution was chosen for biological reasons to allow preference for close neurons. This distribution is superior to the totally random one, but is still not sufficiently robust. Choosing a power law distribution and being careful to making the assignments differently for in and out connectivity proved to be the best. This is thought of as a potentially biological arrangement (Danielle & Bullmore, 2006; Albert-László & Réka, 1999); so LSM style networks with this additional topological constraint can, as of this date, are considered sufficiently biological. Other distributions may also work.

References

Albert, R., Barabási, A.-L.: Topology of evolving networks: local events and universality, vol. 85, pp. 5234–5237 (2000)

Albert-László, B., Réka, A.: Emergence of Scaling in Random Networks. SCIENCE 286, 509–512 (1999)

Bianconi, G., Barabási, A.-L.: Competition and multiscaling in evolving networks. Europhys Lett. 54, 436 (2001)

Chklovskii, L.R.: Structural Properties of the Caenorhabditis elegans Neuronal Network (July 2009)

Danielle, B.S., Bullmore, E.: Small-world brain networks. Neuroscientist 12(6), 512–523 (2006)

Fernando, C., Sojakka, S.: Pattern Recognition in a Bucket. In: Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P., Kim, J.T. (eds.) ECAL 2003. LNCS (LNAI), vol. 2801, pp. 588–597. Springer, Heidelberg (2003)

Gütig, R., Sompolinsky, H.: The tempotron: a neuron that learns spike timing-based decision. Nature Neuroscience 9(3), 420–428 (2006)

Izhikevich, E.M.: Simple Model of Spiking Neurons. IEEE Transactions On Neural Networks 14(6), 1569–1572 (2003)

Jaeger, H.: "The ëcho state" approach to analysing and training recurrent neural networks. German National Research Institute for Computer Science (2001)

Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks. German National Research Institute for Computer Science (2001)

Lukosevicius, M., Jaeger, H.: Reservoir Computing Approaches to Recurrent Neural Network Training. Computer Scinece Review, 127–149 (2009)

Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation 14(11), 2531–2560 (2002)

Maass, W.: Paradigms for computing with Spiking Neurons. Springer, Heidelberg (1999)
Maass, W., Natschlger, T., Markram, H.: Computational models for generic cortical microcircuits. In: Feng, J. (ed.) Computational Neuroscience: A Comprehensive Approach, pp. 575–605. Chapman & Hall/CRC, Boca Raton (2004)
Manevitz, L.M., Marom, S.: Modeling the process of rate selection in neuronal activity. Journal of Theoretical Biology 216(3), 337–343 (2002)
McCullough, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5, 115–127 (1943)
Widrow, B., Hoff, M.: Adaptive Switching Circuits. IRE WESCON Convention Record 4, 96–104 (1960)

A Radial Basis Function Redesigned for Predicting a Welding Process

Rolando J. Praga-Alejo[1], Luis M. Torres-Treviño[2], David S. González[1], Jorge Acevedo-Dávila[1], and Francisco Cepeda[1]

[1] Corporación Mexicana de Investigación en Materiales (COMIMSA)
Calle Ciencia y Tecnología No. 790, Frac. Saltillo 400 C.P. 25290,
Saltillo, Coah., México
Tel.: (844)411-3200 Ext. 1217
{rolandopraga,davidgonzalez,jacevedo,fcocepeda}@comimsa.com
[2] Centro de Innovación, Investigación y Desarrollo en Ingeniería y Tecnología (CIIDIT)
Km. 10 nueva carretera al Aeropuerto Internacional de Monterrey, PIIT Monterrey
Apodaca, Nuevo León, México
luis.torres.ciidit@gmail.com

Abstract. Neural Networks (NNs) have been widely used in many industrial processes for prediction and optimization and they have been proven to be useful tools for explaining complex processes. The main objective of this work consists of improving the accuracy of a Radial Basis Function Neural Network Redesigned by Genetic Algorithm and Mahalanobis distance for predicting a welding process. The evaluation function in this approach considers the use of the Coefficient of Determination R^2. The results indicated that the statistical method R^2 is a good alternative to validate the efficiency of the Neural Network model. The principal conclusion in this work is that the Radial Basis Function Redesigned by Genetic Algorithm and Mahalanobis distance had a very good performance in a real case, considering the prediction of specific responses in a welding process.

Keywords: Radial Basis Function; Genetic Algorithm; Mahalanobis distance; Hybrid Learning; Coefficient of Determination.

1 Introduction

Nowadays, the use of new approaches has been intensified and the companies are supporting in new technologies or numerical methods to improve processes, some of them are intelligent systems and statistical methods [2], [17], [22]. Intelligent systems are derived from different applications or techniques, such as: Fuzzy logic, Neural Networks, Evolutionary Algorithms and Hybrid systems. Intelligent systems simulate or try to imitate human reasoning for solving problems, tasks or decision-making, in complex problems [24], [30]. There are different types of intelligent systems for modeling, predicting and optimizing industrial processes; within these techniques are the Artificial Neural Networks (ANN) [2] [23]. Moreover, also there are different types such as artificial neural network Backpropagation, Support Vector Machines, Adaptive Resonance Theory networks and Radial Basis Function Neural Network (RBFNN) [1], [11], [15], [20], [26]. Neural networks are able to process large amount of information using mathematical model [26]. A Neural Network (NN) is an interconnection of simple processing elements that represents the function of a single neuron. The interconnection is made by an artificial synapse called weights. The adjustment of these weights modifies the performance of the neural network. Neural networks have the ability to approximate functions using huge information from historical or experimental sources [6], [7], [13], [30].

The Radial Basis Function Neural Network (RBFNN) has been developed to predict accurately the performance processes [1]. RBFNN is a second order or hyperspherical type function, the network value represents the distance for a given reference pattern. The Radial Basis Function (RBF) is a hybrid model because it uses both learning supervised and non-supervised. The advantage of these kinds of models is the short time that they need for training. The construction of networks of Radial Basis Function (RBF), in its basic form, includes three totally different layers: input, hidden and output layers (For more details about RBFNN see [1], [10], [13], [15], [20]). The Radial Basis Function Neural Network have problems in two situations, first in the distance given by $\|x-t_i\|^2$ that is called the Euclidean distance, which is composed of the input vector x and the centers t_i. The second difficulty, is the way to calculate the centroids t_i, which are calculated in the different ways, e.g. using the Clustering K-means, through Simulation, Neural Networks or randomly, causing instability to the network.

Moreover, alternative solutions for these problems are the evolutionary algorithms like evolutionary strategies, Genetic Algorithms (GAs). Other alternatives include Particle Swarm Optimization (swarm intelligence), among others. These methods have been successfully used for the selection of the optimal structure of RBFNN.

One of the first works on these applications was realized by [3] where the authors made a training based on a practical data set used to demonstrate the performance of a Genetic Algorithm (GA) in the RBFNN; GA was used to find the hidden layers neurons and centers in the RBFNN. In order to overcome this problem, various methods have been applied to improve the desired parameters and variables through developing models with intelligent systems like [16] that used a technique combining both Genetic Algorithm and RBFNN to determine parameters in this case centers and widths in the network structure. Other similar work was realized by [28], they made a theoretical study of the niche sharing mechanism and also applied it on the performance of a RBF. In [29] applied a similar method with coverage using orthogonal niches to predict better performance on chaotic time series. On the other hand, in [5] proposed another evolutionary algorithm Particle Swarm Optimization that was used to find the hidden layers neurons by means of modifying the centers in the RBFNN, the comparison was made by root mean squared errors. Other work using a Particle Swarm Optimization (PSO) was made by [25] where applied a Quantum-behaved PSO to improve the network parameters. As other application we found the work realized by [27] where they applied the Mahalanobis distance using a GA to find the variance-covariance matrix; this work improves prediction accuracy with symmetrical evolved matrices. Finally it is important to mention the research presented in the article proposed by [10] where a complete description of the development of the radial basis function neural network was made. It becomes giving a description of the structure and functioning of the network, this work mention the Euclidean distances, election of the centers, adjustment of weights, using Gaussians functions among others. This work explains the Gaussian process and its graphical form as well as the local network model; it also explains the similarities and differences between the neural network radial basis, the local network model and the Gaussian process.

The main objective of this work consists in improving the accuracy of the RBFNN using a Hybrid Learning Process with Genetic Algorithm and Mahalanobis distance considering the prediction of specific responses in a real welding process. The RBFNN and Hybrid Learning Process are reviewed in section II and III respectively. In section IV an application is illustrated and finally the results and one discussion are given in section V and VI respectively.

2 Radial Basis Function Neural Network

The RBFNN values represents the distance to a given reference pattern [13]. A RBF has three layers: input, hidden and output. The structure in a RBF, in its basic form, includes three total different layers:

1. Input layer that consists of nodes source x_i (sensory units).
2. The hidden layer is a hidden layer of high dimension and the units (neurons) that form are the basis functions for the input data $\varphi_i(x_i)$.
3. Output layer d_j is the responsible for the activation of patterns considering the input layer network.

The output layer neurons are linear. The hidden layer neurons calculate the difference between the vector of inputs and the weight synapses, called centroid. This difference applies a radial function with shape Gaussian mainly; but it has the advantage of being able to use other radial functions [10]. The function of transfer radial of Gaussian type adopts the following form:

$$G(\|x-t_i\|)=\exp\left(-\|x-t_i\|^2\right). \tag{1}$$

And it is simplified by (2):

$$\varphi_i(x)=G(\|x-t_i\|). \tag{2}$$

Where x_i are the inputs and t_i are the centers or centroid formed by the Euclidean distance or Euclidean norm of $\|x-t_i\|^2$. When it is introducing an input vector, each neuron in the hidden layer has a radial basis function. Each neuron in the hidden layer is the distance between the vector input and its vector of weights. This distance is multiplied by the vector of thresholds or bias [13]. The relationship between inputs and outputs from neural network is given by equation (3) where $y(x_j)=d_j$:

$$y(x) = \sum_{i=1}^{m} wG(\|x - t_i\|) + b \ . \tag{3}$$

The equation is written in matrix form $Gw = d$, the variable d_j is the output or response, $d = [d_1, d_2, \ldots, d_N]^T$ and $w = [w_1, w_2, \ldots, w_m]^T$ which represents the weights determined by equation (4), calculated by Pseudo inverse or by Ordinary Least Square (OLS):

$$\begin{aligned} w &= G^{+}d \\ &= (G'G)^{-1}G'd \end{aligned} \ . \tag{4}$$

There are different types of functions of radial basis, but in this work the function used is given by equation (5) because this function has been developed to predict accurately the performance processes [15] (For more details in Radial functions see [1]).

$$\varphi_i(x) = r^2 \ln(r) \ . \tag{5}$$

Where r is given by the distance between the input variable x and the centroid t_i.

3 Hybrid Learning Process

The purpose of this section is to show a methodology to modify the RBFNN and also improve the parameters and variables that are associated with RBFNN learning processes; in this improvement, it is considering the use of a hybrid intelligent systems and statistical method. We propose to use a Genetic Algorithm and Mahalanobis distance. In addition, the fitness evaluation function is given for the metric R^2, where the objective is to maximize this metric.

3.1 Mahalanobis Distance

The concept of distance for example, given two points x_i, x_j belonging to $\Re^p$, this establishes a distance or a metric among them, if a function d is being defined with the following properties:

- $d : \Re^p \times \Re^p \to \Re^+$, in other words, given two points in the space of dimensions p, its distance with this function is a non-negative number, $d(x_i, x_j) \geq 0$;
- $d(x_i, x_j) = 0$, $\forall i$, the distance between an element and itself is zero.
- $d(x_i, x_j) = d(x_j, x_i)$, the distance is a symmetric function in its arguments.

- $d(x_i, x_j) \leq d(x_j, x_p) + d(x_p, x_j)$, if we have another three points, the sum of the lengths of any two sides of the triangle formed by the three points must always be greater than the third side. This property is known as triangular inequality.

There are different types of distances like: Minkowski distance, the average, the family of metrics Euclidean Weighted and the Euclidean distance which is the most used, e.g. it is applied in the neural network Radial Basis Function. Another measure is the Mahalanobis distance d_m given by equation (6), created by P.C. Mahalanobis (1893-1972), this distance is defined as follows:

$$d_m = \sqrt{(x-\mu)' \Sigma^{-1} \cdot (x-\mu)} \ . \tag{6}$$

The matrix Σ is the covariance matrix and it is square $m \times m$ and symmetric. Note that Mahalanobis distance is distributed as a chi square χ^2 with p degrees of freedom. The probability density function of a χ^2 random variable is $f(x) = \frac{1}{2^{k/2}\Gamma(k/2)} x^{(k/2)-1} e^{-x/2} \quad x > 0$ where k is the number of degrees of freedom. The mean and variance of the χ^2 distribution are k and $2k$ respectively. Note that the chi square random variable is nonnegative and that the probability that the probability distribution is skewed to the right. However, as k increases, the distribution becomes more symmetric. As $k \to \infty$, the limiting form of the chi square distribution is the normal distribution. Define χ^2 as the percentage point or value of the chi square random variable with k degrees of freedom such that the probability that X^2 exceeds this value is α. That is $P(X > \chi^2) = \int_{\chi^2}^{\infty} f(u)du = \alpha$ (see the proof and more details of the Mahalanobis distance in [12] and [21]). For this reason and for its properties, this article uses the Mahalanobis distance. Moreover, the Mahalanobis distance is a distance measure, its utility is that it is a way to determine the similarity between two multidimensional random variables. And it differs from Euclidean distance, because the Mahalanobis distance takes into account the correlation between random variables.

3.2 Genetic Algorithm

The Genetic Algorithms (GAs) are defined as a procedure search based on the mechanisms of genetics and natural selection. These types of algorithms solve difficult problems of a fast, suitable and reliable way [9]. Genetic algorithms (GAs) are the most used evolutionary algorithms. GA was developed by John Holland [14] and has grown as the most used paradigm to solve optimization problems [18]. There are several variants of the GA. Nevertheless, all of them have four general procedures: evaluation of the individuals, selection of the best individuals (in a deterministic or stochastic way), crossover and mutation of individuals and fitness evaluation [8]. Every individual is a solution represented as a binary vector. A set of solutions represents a population of potential solutions or individuals making analogy to natural processes. The GA has binary vectors to represent parameters so real parameters representation requires a decoding procedure to use it in the evaluation procedure. The population begins with random solutions usually with low performance and high diversity. The evolutionary procedures (evaluation, selection, crossover and mutation) are applied of cyclical way, solutions of better performance are obtained and the diversity is lost. It is a more intelligent way to search solutions than "trial and error" procedure. The following steps are involved to generate a near optimal solution, represented by the neural network.

Step 1: Generate a random population.
Step 2: Evaluate every represented solution (a set of parameters) of the population.
Step 3: Select the better evaluated individuals of the population.
Step 4: Generate a new population using crossover and mutation considering the selected individuals.
Step 5: Go to step 1 until an end condition is satisfied.

3.3 Fitness Function and Evaluation

The Fitness Function considers the coefficient of determination R^2, which represents (as regression models) the proportion of variance explained by the model or regressor x. A R^2 closes to one imply that most of the variability of the prediction y, is explained by the model [19].

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \, . \tag{7}$$

Therefore, we need to maximize the metric R^2 given by the equation (7) and R^2 is a global metric evaluation. Where y_i represent the experiment response, $\hat{y}_i$ the neural network predictions or the predicted output and $\bar{y}$ is the mean response [19].

3.4 Radial Basis Function Redesigned

After having identified the problems related to the RBF, some modifications were made to improve its performance. One of these modifications is the calculation of the centroid; this calculation was made randomly and using the method of Clustering K-means for different authors [20]; to replace these methods, another technique is being implemented to calculate centroids, which is derived from the Genetic Algorithm. Another modification to improve the neural network is to apply the Mahalanobis distance instead of the Euclidean distance $\|x - t_i\|^2$. Applying the GA and the

Mahalanobis distance to RBFNN, it rewrites the equation (6) and yields the following:

$$d_m(x, A) = \sqrt{(x - A_i)' \Sigma^{-1} (x - A_i)} \ . \tag{8}$$

Where x represents the input variables of the process, A_i are the centroids calculated by evolutionary algorithm and Σ^{-1} is the inverse of the covariance matrix. As we mentioned, there are different types of Radial Basis Functions, of which we implemented equation (5) in the modified neural network, therefore it is written as follows:

$$\varphi_i(x) = (r)^2 \cdot \ln(r) \qquad r = d_m(x, A) \ . \tag{9}$$

At the same way, the equations (2) and (3) are rewritten as:

$$\varphi_i(x, A) = G(d_m(x, A_i)) \ . \tag{10}$$

$$y(x) = \sum_{i=1}^{n} w \cdot \varphi_i(x, A) + b \ . \tag{11}$$

The following steps are involved to generate the Hybrid Learning Process, in the RBFNN.

Step 1: Generate centroids with GA.
Step 2: Calculate the Mahalanobis distance (Equation (8)).
Step 3: Apply the Radial Basis Function (Equation (9) and (10)).
Step 4: Generate the weights and predictions (Equation (4) and (11)).
Step 5: Evaluate the fitness function in GA (Equation (7)).
Step 6: Go to step 1 until an end condition is satisfied.

4 Application

In order to illustrate the application and efficiency of the hybrid learning process in RBF Redesigned, we applied the proposed RBF to some data of a welding process.

The welding process used was a Laser welding process; in this type of processing, the Laser is used to create a union that has a narrow heat affected zone to reduce the surface roughness of the union and to eliminate the mechanical effects that cause other types of welding processes. Laser systems operating in pulsed or continuous mode can be used for this type of application process. This type of welding has three input variables and one output, mainly, the input variables are: the Width of the spot (which relates to and involves the pulse energy and power density), the other two factors are: Frequency and laser scan Speed were selected for the optimization of the energy consumed per unit of surface area treated. And the response variable for this type of process was the Penetration of the weld in the workpiece. The experimental results are illustrated in Table 1 (for more details about the Laser welding process see [4]).

Table 1. Experimental Data

Parameters			Response
Speed	Width	Frequency	Penetration
1.1	4	3.5	0.6889
1.1	5.6817	3.5	0.2953
1.1	4	3.5	0.6186
1.7727	4	3.5	0.6341
1.5	3	3	0.9280
0.7	5	4	0.4330
0.7	3	3	1.0053
0.7	5	3	0.5568
1.5	5	3	0.5413
1.5	3	4	1.1599
1.1	4	3.5	0.6186
1.1	2.3182	3.5	1.5466
1.1	4	3.5	0.5413
1.1	4	3.5	0.4330
1.1	4	4.3008	0.4794
1.1	4	3.5	0.4952
1.1	4	2.6591	0.4021
0.4272	4	3.5	0.3557
0.7	3	4	1.2218
1.5	5	4	0.3403

In the application, we made a Hybrid Learning Process to optimize the parameters of Radial Basis Function Neural Network with a Genetic Algorithm as a complement. In this model, it is required a solution $A^* = (A_i | i = 1,2,\ldots,m)$ which maximize the metric R^2, so that the set of centers $\{A_i | i = 1,2,\ldots,m\}$ must be determined. The Genetic Algorithm uses a binary representation of the possible solution, this codification is necessary because GA manipulates bits. In the model, a sixteen binary representation per solution was used. The initial population matrix was of 250 rows which represents every individual and $16\times(A_i | i = 1,2,\ldots,m) = 16\times(m\times m)$ matrix to represent every bit of the binary codified solution. That is, every individual of the population represent a set of $(m\times m)$ codified solutions of 16 bits. Tournament selection of size 2 with single point crossover and a simple mutation were used [8]. Probabilistic crossover and mutation of 0.9 and 0.01 were used respectively. The R^2 was used as an evaluation function with 50 generations in the GA.

5 Results

The RBF Redesigned with Evolutionary Algorithm and Mahalanobis distance model, using the data in Table 1 provided a R^2 equal to 94.08.

The results show that RBF Redesigned with Genetic Algorithm and Mahalanobis distance is over 94% in R^2; we refer to R^2 as the amount of variability in the data explained by the models, that is, the RBF Redesigned accounts for more than 94% of

Table 2. Data Predicted

	Parameters			Responses	
Run	Speed	Width	Frequency	Penetration Real	Penetration (Predicted)
1	1.1	4	3.5	0.6889	0.5491
2	1.1	5.6817	3.5	0.2953	0.3576
3	1.1	4	3.5	0.6186	0.5491
4	1.7727	4	3.5	0.6341	0.6324
5	1.5	3	3	0.9280	0.9146
6	0.7	5	4	0.4330	0.3597
7	0.7	3	3	1.0053	0.9107
8	0.7	5	3	0.5568	0.4768
9	1.5	5	3	0.5413	0.5138
10	1.5	3	4	1.1599	1.1607
11	1.1	4	3.5	0.6186	0.5491
12	1.1	2.3182	3.5	1.5466	1.6008
13	1.1	4	3.5	0.5413	0.5491
14	1.1	4	3.5	0.4330	0.5491
15	1.1	4	4.3008	0.4794	0.5511
16	1.1	4	3.5	0.4952	0.5491
17	1.1	4	2.6591	0.4021	0.4959
18	0.4272	4	3.5	0.3557	0.5126
19	0.7	3	4	1.2218	1.1904
20	1.5	5	4	0.3403	0.3237

the variability in the data. The metric is the quantity used to express the proportion of total variability in the response accounted by the model. So that R^2 indicates the proportion of variability in $\bar{y}$ explained by the model. If the R^2 value is very close to 100% or above 80%, it means that the model will be a good predictor [19].

The prediction of the data with the RBF Redesigned model shown in Table 2, and the data had an appropriate performance with the RBF network modified, for validate the model, a testing was performed with the data and parameters of the run 19 (Table 2) since they are the values with better penetration and better quality in the depth of the weld, and to see the performance of penetration and optimization of the proposed neural network. The parameters were Speed 0.7, Width 3 and Frequency 4. The result and final test was 1.2232 penetration in the real case, as shown in Figure 1, versus 1.1904 (Table 2) prediction.

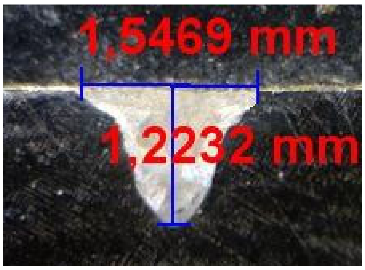


Fig. 1. Laser penetration welding process (Test)

6 Discussion

The Hybrid Learning Process method proposed in this work, it is applied a Genetic Algorithm and Mahalanobis distance, instead of computing the centers matrix by Genetic Algorithm, it is determined in such a way that maximizes the Coefficient of determination R^2 and the Fitness Function depends on the prediction accuracy fitted by the hybrid learning approach, where the Coefficient of determination R^2 is a global metric evaluation. And the Mahalanobis distance is a distance measure, which it uses the correlation between variables and it takes the covariance and variance matrix in the input variables; its utility is that it is a way to determine the similarity between two variables in this case between x and A_i; where x represents the input variables, A_i are the centroids calculated by a Genetic Algorithm. And this distance helps in reduce the variance into variables. Note that while R^2 is near to 100%, this indicates that the model is good for predicting and optimizing, the metric R^2, which tells us that if we have values above 80% will be good model to predict and optimize [19]. The metric R^2 with RBF Redesigned is 94.08; it indicates that the proposed model is a good method to predict a Laser welding process.

As a conclusion, we can say that the neural network Redesigned in this work had a good performance, since the method of Hybrid Learning Process presented in this work applies a Genetic Algorithm to calculate the matrix of centers, where the Coefficient of determination R^2, it becomes the statistical evaluation function of Genetic Algorithm, which helps the accuracy of the prediction and optimization of the network of Radial Basis Function. And with these modifications, the contribution and the difference that exists in this work with others authors, it is that the method of Hybrid Learning Process not needs to calculate and aggregate the widths of the hidden layers of the RBF network. Hence, the method eliminate the way to get the centroids and Euclidean distance, which causing problems in the predictions. These problems are solved first, applying the Mahalanobis distance that takes into account the correlation between variables and secondly finding the centers with Genetic Algorithms considered in the evaluation function the Coefficient of determination R^2, that it is a good alternative method to validate the efficiency of the Neural Network model; where the GA finds the optimal centroids for improving the accuracy of the RBFNN; the form of parallel implicit search that has the GA stands out on other paradigms (Evolutionary Algorithms) that are much simpler than the GA; for this reason in this work the GA was in use. And the model proposed is a good tool for prediction in a real case (see Fig. 1).

References

1. Arbib Michael, A.: The Handbook of Brain Theory and Neural Networks. The MIT Press, London (2003)
2. Benyounis, K.Y., Olabi, A.G.: Optimization of different welding process using statistical and numerical approaches – A reference guide. Advances in Engineering Software 39, 483–496 (2008)

3. Billings Steve, A., Zheng Guang, L.: Radial Basis Function Network Configuration Using Genetic Algorithms. Neural Networks 8(6), 877–890 (1995)
4. Cepeda, F.: Análisis y optimización de parámetros empleados en la unión y tratamientos superficiales mediante laser CO_2 y Nd-YAG en la aleación de cobalto ASTM F-75. Master Thesis. COMIMSA (2010)
5. Ding, H., Xiao, Y., Yue, J.: Adaptive Training of Radial Basis Function Networks using Particle Swarm Optimization Algorithm. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3610, pp. 119–128. Springer, Heidelberg (2005)
6. Freeman, J.A., Skapura, D.M.: Redes Neuronales. In: Algoritmos, aplicaciones y técnicas de propagación, p. 306. Addison-Wesley, México (1993)
7. Freeman, J.A., Skapura, D.M.: Neural Networks. In: Algorithms, applications, and Programming Techniques, Addison-Wesley, Reading (1991)
8. Goldberg David, E.: Genetic algorithms in search, optimization, and machine learning. Oxford University Press, New York (1989)
9. Goldberg David, E.: The design of Innovation, Genetic Algorithms and Evolutionary Computation. Kluwer Academic Publishers, USA (2002)
10. Gregorcic, G., Lightbody, G.: Nonlinear system identification: From multiple-model networks to Gaussian processes. Engineering Applications of Artificial Intelligence (2008)
11. Hagan, Demuth, Beale: Neural Network Design. Thomson Publishing Inc., USA (1996)
12. Härdle, W., Simar, L.: Applied Multivariate Statistical Analysis, 2nd edn. Springer, Heidelberg (2007)
13. Haykin, S.: Neural Networks: A comprehensive foundation. Prentice-Hall, Englewood Cliffs (1999)
14. Holland John, H.: Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor (1975)
15. Martín del Brío, B., Sanz, A.: Redes Neuronales y Sistemas Difusos. Alfaomega. España. (2007)
16. Meqin, L., Jida, C.: Determining the structures and parameters of Radial Basis Function Neural Networks using improved Genetic Algorithms. Journal CSUT 5(2), 141–146 (1998)
17. Meziane, F., Vadera, S., Kobbacy, K., Proudlove, N.: Intelligent systems in manufacturing: current developments and future prospects. Journal Integrated Manufacturing Systems 11(4), 218–238 (2000)
18. Mitsuo, G., Chen, R.: Genetic Algorithms & Engineering Optimization. John Wiley & Sons, Chichester (2000)
19. Montgomery, D.C., Peck, E.A., Vining, G.G.: Introducción al Análisis de Regresión Lineal. In: Editorial Continental, Tercera edición, México (2006)
20. Nelles, O.: Nonlinear System Identification: From classical approaches to neural networks and fuzzy models. Springer, Heidelberg (2001)
21. Peña, D.: Análisis de Datos Multivariantes. Mc Graw Hill, New York (2002)
22. Praga-Alejo, R., Torres-Treviño, L., Piña-Monarrez, M.: Prediction in Welding Process Using Multiple Linear Regression and Neural Network. International Journal of Industrial Engineering, 481–488 (2008) ISSN 1072-4761
23. Praga-Alejo, R., Torres-Treviño, L., Piña-Monarrez, M.: Comparison between several neural networks models using statistical methods. In: Proceedings of the 13th Annual International Conference on Industrial Engineering Theory, Applications and Practice, Las Vegas, Nevada (September 7-10, 2008)

24. Rao, S., Nahm, A., Shi, Z., Deng, X., Syamil, A.: Artificial intelligence and expert systems applications in new product development. Kluwer Academic Publishers, USA (1999)
25. Sun, J., Xu, W., Liu, J.: Training RBF Neural Network via Quantum-Behaved Particle Swarm Optimization. In: King, I., Wang, J., Chan, L.-W., Wang, D. (eds.) ICONIP 2006. LNCS, vol. 4233, pp. 1156–1163. Springer, Heidelberg (2006)
26. Tsoukalas, L., Uhrig, R.: Fuzzy and neural approaches in engineering. Wiley-Interscience Publication, USA (1998)
27. Valls, J.M., Aler, R., Fernández, O.: Using a Mahalanobis-Like Distance to train Radial Basis Neural Networks. In: Cabestany, J., Prieto, A.G., Sandoval, F. (eds.) IWANN 2005. LNCS, vol. 3512, pp. 257–263. Springer, Heidelberg (2005)
28. Whitehead Bruce, A., Choate Timothy, D.: Cooperative – Competitive Genetic Evolution of Radial Basis Function Centers and Widths for Time Series Prediction. IEEE Transactions on Neural Networks (1995)
29. Whitehead Bruce, A.: Genetic Evolution of Radial Basis Function Coverage using Orthogonal Niches. IEEE Transactions on Neural Networks 7, 869–880 (1996)
30. Yen, J., Langari, R.: Fuzzy Logic: Intelligence, Control and Information. Prenhall, USA (2000)

Dynamic Neural Networks Applied to Melody Retrieval

Laura E. Gomez, Humberto Sossa, Ricardo Barron, and Julio F. Jimenez

Centro de Investigación en Computación-IPN,
Unidad Profesional Adolfo-López Mateos,
Av. Juan de Dios Bátiz s/n and M. Othón de Mendizábal,
Zacatenco, México, DF. 07738, Mexico
sgomezb08@sagitario.cic.ipn.mx, hsossa@cic.ipn.mx,
rbarron@cic.ipn.mx, jfvielma@cio.mx

Abstract. A new method for the retrieval of melodies from a database is described in this paper. For its functioning, the method makes use of Dynamic Neural Networks (DNN). During training a set ofDNN is first trained with information of the melodies to be retrieved. Instead of using traditional signal descriptors we use the matrix of synaptic weights that can be efficiently used for melody representation and retrieval. Most of the reported works have been focused on the symbolic representation of musical information. None of them have provided good results with original signals.

Keywords: Music Information Retrieval; Dynamic Neuronal Networks; Musical Descriptors.

1 Introduction

With the explosive expansion of digital music and audio contents, efficient retrieval of such data is getting more and more attention, especially in large-scale multimedia database applications. In the past, music information retrieval was based on textual metadata such as title, composer, singer or lyric. However, these various metadata-based schemes for music retrieval have suffered from many problems including extensive human labor, incomplete knowledge and personal bias.

Compared with traditional keyword-based music retrieval, content-based music retrieval provides more flexibility and expressiveness. Content-based music retrieval is usually based on a set of extracted music features such as pitch, duration, and rhythm.

In some works, such as in [1] and [2], only pitch contour is used to represent melodies. Music melody is transformed into a stream of U, D, R, which stands for a note higher than, lower than, or equal to the previous note, respectively. This method simplifies the melody so much that it cannot discriminate among melodies very well among melodies, especially when the music contains a lot of data.

In order to represent the melody more accurately and discriminatively, new feature sets have been proposed. In [3], pitch interval and rhythm are considered as well as pitch contour. In [4], relative interval slope is used in music information retrieval, while in [5] four basic segment types (A,B,C,D) to model music contour are introduced.

When rhythm and pitch interval are considered, more complex similarity measures and matching algorithms should be used. In [5], the authors use a two-dimensional augmented suffix trees to search the desired song, rather than approximate string matching algorithm used in [1] and [2]. In [6], a new distance metric between query and songs is proposed. Its computation is very time-consuming for many parameters need to be adjusted to find the minimum distance.

Among neural networks, dynamic networks such as Hopfield networks, Jordan and Elman [7] have been extensively used. On the other hand, multilayer neural networks, static in nature can achieve a dynamic behavior by reinforcing their own inputs samples of their previous values.

In this paper, we describe a novel music retrieval proposal based on the use of dynamic neural networks (DNN). The idea is to put into operation a system for music retrieval. DNN are trained with samples of the melodies to be retrieved. We then use the synaptic weights of the DNN as descriptors for the recovery of the melody.

The rest of this paper is organized as follows. In Section 2, we present an overview of ongoing research for analyzing music features and constructing MIR systems. In Section 3, we describe our music retrieval system using dynamic neural networks. In Section 4, we report and discuss some of the experimental results obtained. In section 5, we finally conclude and describe future directions for research.

2 Related Work

In this section, we review some of the typical techniques and reported systems for music information retrieval. As we know, music can be represented in two different ways. One is based on musical scores such as MIDI and Humdrum [8]. The other is based on acoustic signals which are sampled at a certain frequency and compressed to save space. Wave **(.wav)** and MPEG Layer-3 **(.mp3)** are examples of this representation.

2.1 Symbolic Analysis

Many research efforts to solve the music similarity problem have used symbolic representation: MIDI, musical scores, note lists and so on. Based on this, pitch tracking finds a ''melody contour'' for a piece of music. Next, a string matching technique can be used to compare the transcriptions of songs. Refer for example to [1], [9], [10], [11] and [12].

String matching has been widely used in music retrieval due to melodies are represented using a string sequence of notes. To consider human input errors, dynamic programming has been applied to the string matching; however, this method tends to be time consuming. An inexact model matching approach reported in [13] is based on a quantified inexact signature-matching approach to find an approximate model to users' query requirements. It can enhance the reusability of a model repository and makes it possible to use and manage a model repository conveniently and flexibly. Zhuge tried to apply this theory to a problem-oriented model repository system PROMBS [14].

There are also researches for symbolic MIR based on the ideas from traditional text IR. Using traditional IR techniques such as probabilistic modeling is described in [15], and using approximate string matching in [16]. Some work addressed other IR issues such as ranking and relevance. Hoashi [17] used relevance feedback for music retrieval based on the tree-structured vector quantization method (TreeQ) developed by Foote. The TreeQ method trains a vector quantizer instead of modeling the sound data directly.

2.2 Acoustic Signal Analysis

There are many techniques to extract pitch contour, pitch interval, and duration from a voice humming query. In general, methods for detecting pitches can be divided roughly into two categories: time-domain based and frequency-domain based.

In the time-domain, ZCR (zero crossing rate) and ACF (auto correlation function) are two popular methods. The basic idea is that ZCR gives information about the spectral content waveform cross zero per unit time [18]. In recent works, ZCR appeared in a different form such as VZCR (variance of ZCR) or SZCR (smoothing ZCR) [19]. On the contrary, ACF is based on the cross correlation function. While a cross correlation function measures the similarity between two waveforms along the time interval, ACF can compare one waveform with itself.

In the frequency-domain, FFT (Fast Fourier Transformation) is one of the most popular methods. This method is based on the property that every waveform can be divided into simple sine waves. But, a low spectrum rate for longer window may increase the frequency resolution while decreasing the time resolution. Another problem is that the frequency bins of the standard FFT are linearly spaced, while musical pitches are better mapped on a logarithmic scale. So, Forberg [20] used an alternative frequency transformation such as constant Q transform spectrums which are computed from tracked parts.

In recent works for the automatic transcription, they used probabilistic machine learning techniques such as HMM (Hidden Markov Models) and ANN (Artificial Neural Networks) to identify salient audio features to reduce the dimensionality of feature space. Ryynanen and Klapuri [21] proposed a singing transcription system based on the HMM-based notes event modeling. The system performed note segmentation and labeling and also applied multiple-F0 estimation method [22] for calculating the fundamental frequency.

2.3 Recent MIR Systems

For decades, many researchers have developed content based MIR (Music Information Retrieval) systems based on both acoustic and symbolic representations, refer for example to [1], [9], [23] and [12].

Ghias [1] developed a QBH system capable of processing acoustic input in order to extract appropriate query information. However, this system used only three types of contour information to represent melodies. The MELDEX system [9] was designed to retrieve melodies from a database using a microphone. It first transformed acoustic query melodies into music notations; then it searched the database for tunes containing the hummed (or similar) pattern. This web-based system provided several match modes including approximate matching for interval, contour, and rhythm.

MelodyHound [23], originally known as the ‘‘TuneServer’’, also used only three types of contour information to represent melodies. They recognized the tune based on error-resistant encoding. Also, they used the direction of the melody only, ignoring the interval size or rhythm. The C-BRAHMS [24] project developed nine different algorithms known as P1, P2, P3, MonoPoly, IntervalMatching, PolyCheck, Splitting, ShiftOrAnd, and LCTS for dealing with polyphonic music.

Suzuki [25] proposed a MIR system that uses both lyrics and melody information in the singing voice. They used a finite state automaton (FSA) as a lyric recognizer to check the grammar and developed an algorithm for verifying a hypothesis output by a lyric recognizer. Melody information is extracted from an input song using several pieces information of hypothesis such as song names, recognized text, recognition score, and time alignment information.

Many other researchers have studied quality of service (QoS)-guaranteed multimedia systems over unpredictable delay networks by monitoring network conditions such as available bandwidth. McCann [26] developed an audio delivery system called Kendra that used adaptability with a distributed caching mechanism to improve data availability and delivery performance over the Internet. Huang [27] presented the PARK approach for multimedia presentations over a best-effort network in order to achieve reliable transmission of continuous media such as audio or video.

3 Dynamic Neuronal Networks Applied to MIR

Dynamic neural networks are an extension of static neural networks via the consideration of time. The proposed dynamic models are developed based on static MLFN. In general, dynamics can be expressed by using a tapped-delay line, external dynamics and internal dynamics [28]. Tapped-delay line approach uses a sequence of delay to express dynamics and forms time-delay neural network [29] and [30]. So called external dynamic approach uses the historical information of output itself to get a kind of autoregressive type neural network [31] and [32].

3.1 Time Delay Neural Network

Time delay neural networks (TDNN) arise as an extension of static neural networks, which are designed to explicitly include time relationships between input-output mappings. Time-lagged feedforward networks (TLFN) are a special type of dynamic networks that integrate linear filter structures inside a feedforward neural network to extend the non-linear mapping capabilities of the network with a representation of time [33]. Thus, in TLFN the time representation is brought inside the learning machine. The advantage of this technique is that the learning machine can use filtering information while the disadvantage is that the learning becomes complex since the time information is also coded in. TDNN is one of the specific cases of TLFN where a tapped delay line is given in the input followed by a multilayer perceptron (MLP) as shown in Figure 1.

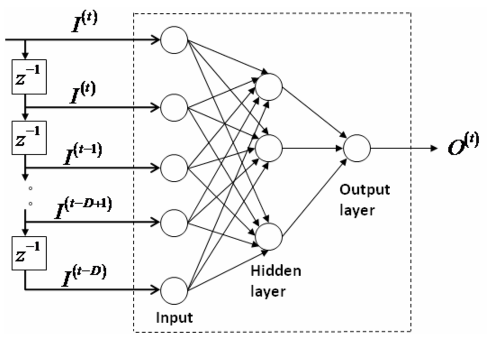


Fig. 1. Structure of a Time Delay Neural Network

Current input (at time t) and D delayed inputs (at time $t-1, t-2, ..., t-D$) can be seen by the TDNN. The TDNN can be trained by using gradient descent back propagation. The ordered training patterns must be provided during the training process [34].

3.2 Proposed Method

For training we used WAV files of the melodies. The set of melodies from a database is used to train a dynamic neural network (TDNN). This is shown in Figure 2. At the end of the training stage, the matrices of weights of the NN are obtained. Each set of weights (WNN_i) is used as a descriptor of a trained melody.

This method is novel because it works in the time domain, not in the frequency domain which gives a digital signature, such as: 1) music features: pitch, duration, and rhythm, or 2) traditional descriptors: pitch contour, zero crossing rate, cross correlation, FFT, and others.

So, instead of using traditional digital signatures or features for a melody, we use the information directly from the melody signal. This reduces the level of a-priori knowledge of the melody by the user.

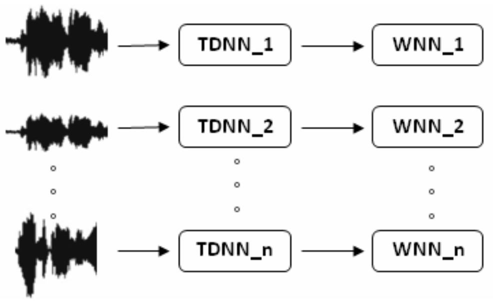


Fig. 2. Structure training of the TDNN with melodies

Recovery of a melody is performed by means of a segment of the melody as a query. This segment is processed by the bank of already trained TDNN with the synaptic weights (WNN_i) obtained during training to obtain the error rate recovery (Re_i). Finally we use the argument of the minimum to obtain the index of the melody that contains the query input. This procedure is depicted in Figure 3.

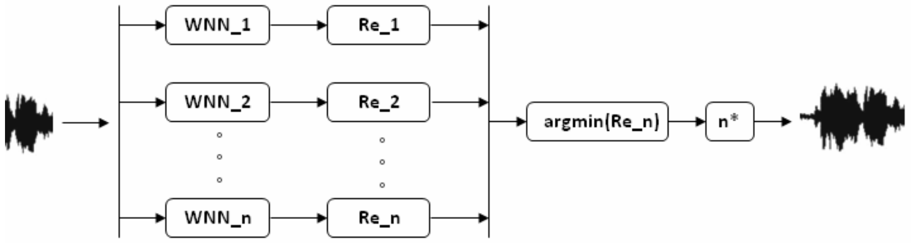


Fig. 3. Retrieval procedure of a melody using the proposed model

Error recovery is given by equation (1), where x_i are the trained matrices of weights, y_i is the segment to recognize, and w is the number of windows in which the segment was divided.

$$\mathrm{Re\_}i = \frac{\sum_{j=1}^{N}\left(x_j - y_j\right)^2}{w} \tag{1}$$

A problem is that input pattern has to compared will all the ANNs. A solution could be to use a distributed and parallel computing paradigm; Figure 4 shows a basic "family tree" for a parallel computer architecture.

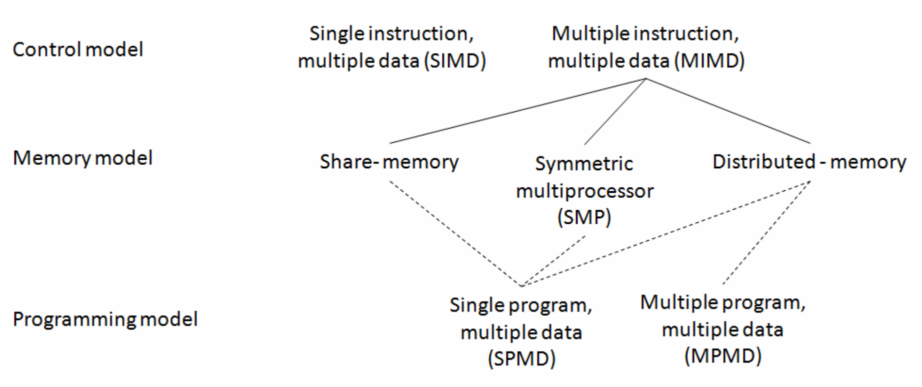

Control model
Single instruction, multiple data (SIMD)
Multiple instruction, multiple data (MIMD)
Memory model
Share-memory
Symmetric multiprocessor (SMP)
Distributed-memory
Programming model
Single program, multiple data (SPMD)
Multiple program, multiple data (MPMD)

Fig. 4. "Genealogy" of parallel computing systems

The *control model* dictates how many different instructions can be executed simultaneously. The terms SIMD (Single instruction, multiple data) and MIMD (multiple instruction, multiple data) date from parallel computing's early days; booth are still in use although no longer the only being a distinguishing feature of a parallel computer.

The *Memory model* indicates how many CPUs can directly access a given memory location. All CPUs access a single memory in shared-memory computers, whereas distributed-memory computers use a separate memory for each CPU.

The *programming model* refers to the restrictions concerning the number of executables that can participate in a parallel execution. In multiple-program, multiple-data model, the programmer creates a separate executables for each CPU; for the single-program, multiple-data model, all instructions to be carried out by the all CPUs are combined into a single executable.

4 Experimental Results

Either for training or for retrieval, we used 16-bit WAV files (in stereo mode). One file was used to train a DNN. A maximum of 50 training iterations was used, and a maximum of 50 neurons were adopted for the hidden layer. At the end of training the corresponding weight matrices are obtained, which are used as descriptors for the learned melodies.

Tests were performed with different numbers of neurons in the hidden layer, as well as different numbers of iterations. Using a database of Disney, the results of these tests are shown in Tables 1 and 2 and Figures 5 and 6, the error rate training is obtained with averaging errors of each TDNN, shown equation (2).

$$\varepsilon = \frac{\sum_{i=1}^{N} (e_i)}{N} \tag{2}$$

where e_i is the training error of each TDNN, and ε is the total error rate training, theses shown in Table 1, and using the equation (1) shows in the Table 2.

Through experimentation, we have observed that the system works effectively with a query composed of less than 1 percent of the total melody. In all cases, the melody was correctly recovered.

Table 1. Table of error rate training, with different numbers of neurons and iterations

	10 Iterations			30 Iterations			50 Iterations		
N. neurons	Minimum	Average	Maximum	Minimum	Average	Maximum	Minimum	Average	Maximum
5	1.36E-03	6.50E-03	1.10E-02	6.45E-04	5.55E-03	1.05E-02	3.19E-04	5.37E-03	1.07E-02
10	1.68E-03	5.71E-03	1.06E-02	3.63E-04	5.43E-03	1.05E-02	3.00E-04	5.17E-03	1.04E-02
15	3.05E-04	5.26E-03	1.05E-02	3.52E-04	5.17E-03	1.06E-02	2.87E-04	5.23E-03	1.06E-02
20	3.80E-04	5.26E-03	1.04E-02	2.87E-04	5.19E-03	1.04E-02	3.05E-04	5.00E-03	1.04E-02
25	2.90E-04	5.26E-03	1.05E-02	2.85E-04	5.14E-03	1.04E-02	2.94E-04	4.95E-03	1.02E-02
30	2.85E-04	5.22E-03	1.04E-02	2.99E-04	4.98E-03	1.01E-02	2.92E-04	5.01E-03	1.04E-02
35	2.82E-04	5.32E-03	1.04E-02	2.85E-04	**4.92E-03**	1.02E-02	2.88E-04	4.97E-03	1.02E-02
40	**2.79E-04**	5.62E-03	**1.03E-02**	2.82E-04	4.97E-03	1.02E-02	**2.76E-04**	4.96E-03	1.01E-02
45	3.17E-04	5.04E-03	1.03E-02	2.84E-04	5.08E-03	1.03E-02	2.88E-04	4.94E-03	1.01E-02
50	2.84E-04	**5.03E-03**	1.03E-02	**2.80E-04**	4.96E-03	**1.01E-02**	2.80E-04	**4.86E-03**	**1.01E-02**

Table 2. Table of error rate recovery, with different numbers of neurons and iterations

	10 Iterations			30 Iterations			50 Iterations		
N. neurons	Minimum	Average	Maximum	Minimum	Average	Maximum	Minimum	Average	Maximum
5	5.96E-03	2.32E-02	5.72E-02	**5.49E-03**	1.51E-02	2.30E-02	1.03E-02	**1.47E-02**	**2.21E-02**
10	**5.54E-03**	1.68E-02	2.59E-02	5.92E-03	2.15E-02	4.97E-02	8.93E-03	1.73E-02	2.99E-02
15	9.41E-03	1.69E-02	3.14E-02	6.21E-03	1.55E-02	2.50E-02	9.27E-03	2.01E-02	4.41E-02
20	9.56E-03	1.90E-02	4.52E-02	9.33E-03	2.18E-02	5.81E-02	**8.02E-03**	2.12E-02	5.54E-02
25	8.93E-03	1.70E-02	3.98E-02	9.45E-03	**1.41E-02**	**2.23E-02**	8.75E-03	1.53E-02	2.37E-02
30	9.44E-03	1.97E-02	4.85E-02	8.31E-03	2.01E-02	3.78E-02	8.32E-03	2.04E-02	5.32E-02
35	1.07E-02	2.89E-02	7.97E-02	9.59E-03	1.49E-02	2.44E-02	9.50E-03	2.26E-02	6.09E-02
40	1.26E-02	1.94E-02	3.58E-02	9.38E-03	2.07E-02	5.26E-02	1.23E-02	1.87E-02	3.75E-02
45	8.95E-03	1.76E-02	3.79E-02	9.26E-03	1.84E-02	4.36E-02	9.05E-03	1.73E-02	3.57E-02
50	9.31E-03	**1.44E-02**	**2.08E-02**	1.11E-02	1.64E-02	2.74E-02	9.72E-03	1.81E-02	3.40E-02

If for example, if we train the TDNN with a melody composed of 7,826,688 frames, we observed that with only 20,717 frames the melody was correctly retrieved. This represents 0.264 percent of the total of the melody. With this we can conclude that recovering can be performed in this particular case with less than 1 percent of information of the melody.

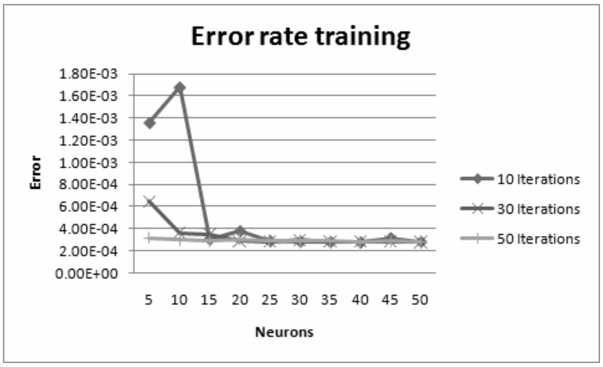


Fig. 5. Graphic of rate training, with different numbers of neurons and iterations

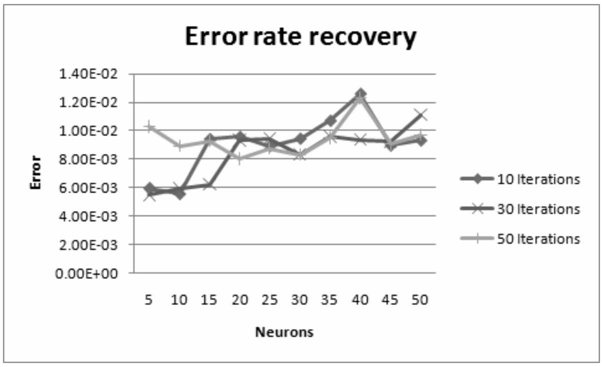


Fig. 6. Graphic of rate recovery, with different numbers of neurons and iterations

5 Conclusion and Present Research

Content-based music retrieval is a very promising method for large music library searching, yet it is a very challenging problem. We discussed various issues for content-based retrieval. In this paper, we have presented a preliminary approach for Music Information Retrieval. The goal of this study was to explore a new line of research within the field of MIR. Not all people are experts in models auditory perception, so we have chosen a TDNN network type that is capable of solving the problem from the samples without using any descriptor or digital signature. Neither any preprocessing to the music files was performed. This can modify the melody content or even distort the information of the melody. Therefore, unlike other MIR reported techniques, we introduce the original melody directly into the net.

The input to the bank of TDNN can be considered as a time series of the amplitudes from the melody. The output of the network encodes a description of the melody in the matrix of weights.

Through experimentation, we have observed that the system performs very well with queries composed with less than 1 percent of the total melody. In all cases, the corresponding melody was correctly recovered.

In short, it can be concluded that a system retrieval based on the functioning of DNN could be a good option for MIR.

Nowadays, we are doing the following. 1) We are going to increase the database of melodies to several thousands of samples to verify if the performance of the proposal continues to be as good as now, and 2) We are going to experiment with different kind of noises (additive, subtractive, mixed, Gaussian) to study the behaviour of the proposal. Once we have done this, we are going to test and if necessary make the corresponding modifications to the proposal when the queries are composed of parts of different versions of the melodies used for training, for example with queries composed of parts of a melody but sang by people, this, of course represents the most difficult challenge to face. We are also planning to use associative memories to accomplish the same task.

Acknowledgements. We wish to thank the Centro de Investigación en Computación of the I.P.N. by the support to accomplish this project. L.E. Gomez and J.F. Jimenez thank CONACYT by the scholarship received to complete their doctoral studies. R. Barron thanks the SIP-IPN under grant 20100379 for the support. H. Sossa thanks the SIP-IPN under grant 20100468 for the support. Authors thank the European Union, the European Commission and CONACYT for the economical support. This paper has been prepared by economical support of the European Commission under grant FONCICYT 93829. The content of this paper is an exclusive responsibility of the CIC-IPN and it cannot be considered that it reflects the position of the European Union. Finally, authors thank the reviewers for their comments for the improvement of this paper.

References

[1] Ghias, A.: Query By Humming-Musical Information Retrieval in an Audio Database. Proc.s of ACM Multimedia 95, 231–236 (1995)

[2] Blackburn, S., De Roure, D.: A Tool for Content Based Navigation of Music. Proc. ACM Multimedia 98, 361–368 (1998)

[3] McNab, R.J.: Towards the Digital Music Library: Tune Retrieval from Acoustic Input. In: Proc. of Digital Libraries, pp. 11–18 (1996)

[4] Lemstrom, K., Laine, P., Perttu, S.: Using Relative Interval Slope in Music Information. Retrieval. In: Proc. of International Computer Music Conference 1999 (ICMC 1999), pp. 317–320 (1999)

[5] Chen, A.L.P., Chang, M., Chen, J.: Query by Music Segments: An Efficient Approach for Song Retrieval. In: Proc. of IEEE International Conference on Multimedia and Expo. (2000)

[6] Francu, C., Nevill-Manning, C.G.: Distance Metrics and Indexing Strategies for a Digital Library of Popular Music. In: Proc. of IEEE International Conference on Multimedia and Expo. (2000)

[7] Acosta, M., Salazar, H., Zuluaga, C.: Tutorial de Redes Neuronales, Universidad Tecnológica de Pereira (2000)

[8] Kornstadt, A.: Themefinder: A web-based melodic search tool. In: Computing in Musicology, vol. 11, MIT Press, Cambridge (1998)

[9] McNab, R.J., et al.: The New Zealand digital library melody index. Digital Libraries Magazine (1997)

[10] Uitdenbogerd, A., Zobel, J.: Melodic matching techniques for large music databases. In: Proceedings of ACM Multimedia Conference, pp. 57–66 (1999)

[11] Hwang, E., Rho, S.: FMF(fast melody finder): A web-based music retrieval system. In: Wiil, U.K. (ed.) CMMR 2003. LNCS, vol. 2771, pp. 179–192. Springer, Heidelberg (2004)

[12] Hwang, E., Rho, S.: FMF: Query adaptive melody retrieval system. Journal of Systems and Software 79(1), 43–56 (2006)

[13] Zhuge, H.: An inexact model matching approach and its applications. Journal of Systems and Software 67(3), 201–212 (2003)

[14] Zhuge, H.: A problem-oriented and rule-based component repository. Journal of Systems and Software 50(3), 201–208 (2000)

[15] Pickens, J.: A comparison of language modeling and probabilistic text information retrieval approaches to monophonic music retrieval. In: Proceedings of the 1st Annual International Symposium on Music Information Retrieval, ISMIR 2000 (2000)

[16] Lemstrom, K., Wiggins, G.A., Meredith, D.: A threelayer approach for music retrieval in large databases. In: Second International Symposium on Music Information Retrieval, Bloomington, IN, USA, pp. 13–14 (2001)

[17] Hoashi, K., Matsumoto, K., Inoue, N.: Personalization of user profiles for content-based music retrieval based on relevance feedback. ACM Multimedia, 110–119 (2003)

[18] Gerhard, D.: Pitch Extraction and Fundamental Frequency: History and Current Techniques. Technical Report TR-CS 2003-06 (2003)

[19] Huang, R., Hansen, J.H.L.: Advanced in unsupervised audio classification and segmentation for the broadcast news and NGSW Corpora. IEEE Trans. on Audio Speech and Language Processing 14(3), 907–919 (2006)

[20] Forberg, J.: Automatic conversion of sound to the MIDIformat. TMH-QPSR 1-2/1998 (1998)

[21] Ryynanen, M., Klapuri, A.: Transcription of the singing melody in polyphonic music, ISMIR 2006 (2006)

[22] Klapuri, A.P.: A perceptually motivated multiple-f0 estimation method. In: 2005 IEEE workshop on applications of signal processing to audio and acoustics, pp. 291–294 (2005)

[23] Typke, R., Prechelt, L.: An interface for melody input. ACM Transactions on Computer–Human Interaction, 133–149 (2001)
[24] Ukkonen, E., Lemstrom, K., Makinen, V.: Sweepline the music. In: Klein, R., Six, H.-W., Wegner, L. (eds.) Computer Science in Perspective. LNCS, vol. 2598, pp. 330–342. Springer, Heidelberg (2003)
[25] Suzuki, M., et al.: Music information retrieval from a singing voice based on verification of recognized hypothesis. In: ISMIR 2006 (2006)
[26] McCann, J.A., et al.: Kendra: Adaptive Internet system. Journal of Systems and Software 55(1), 3–17 (2000)
[27] Huang, C.M., et al.: Synchronization and flow adaptation schemes for reliable multiple-stream transmission in multimedia presentation. Journal of Systems and Software 56(2), 133–151 (2001)
[28] Nelles, O.: Nonlinear system identification. Springer, Heidelberg (2001)
[29] Yun, S.Y., Namkoong, S., Rho, J.H., Shin, S.W., Choi, J.U.: A performance evaluation of neural network models in traffic volume forecasting. Mathematic Computing Modelling 27(9-11), 293–310 (1998)
[30] Lingras, P., Mountford, P.: Time delay neural networks designed using genetic algorithms for short term inter-city traffic forecasting. In: Monostori, L., Váncza, J., Ali, M. (eds.) IEA/AIE 2001. LNCS (LNAI), vol. 2070, Springer, Heidelberg (2001)
[31] Campolucci, P., Uncini, A., Piazza, F., Rao, B.D.: On-line learning algorithms for locally recurrent neural networks. IEEE transactions on neural networks 10(2), 253–271 (1999)
[32] Tsai, T.-H., Lee, C.-K., Wei, C.-H.: Artificial Neural Networks Based Approach for Short-term Railway Passenger Demand Forecasting. Journal of the Eastern Asia Society for Transportation Studies 4, 221–235 (2003)
[33] Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. p. 837. Prentice Hall PTR, Englewood Cliffs (1998) ISBN 0-13-273350-1
[34] Weibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K.: Phenomena Recognition Using Time-delay Neural Networks. IEEE Transactions on Acoustics, Speech, and Signal Processing 37, 328–339 (1989)

Recognition of Huffman Codewords with a Genetic-Neural Hybrid System

Eugène C. Ezin[1], Orion Fausto Reyes-Galaviz[2], and Carlos A. Reyes-García[3]

[1] Université d'Abomey-Calavi
Institut de Mathématiques et de Sciences Physiques
Unité de Recherche en Informatique et Sciences Appliquées
BP 613 Porto-Novo, Rep. Bénin
eugene.ezin@imsp-uac.org

[2] Universidad Autónoma de Tlaxcala, Facultad de
Ciencias Básicas, Ingeniería y Tecnología
orionfrg@yahoo.com

[3] Instituto Nacional de Astrofísica, Óptica y Electrónica
Luis Enrique Erro 1, Tonantzintla, Puebla, México
kargaxxi@inaoep.mx

Abstract. Character classification is known to be one of many basic applications in the field of artificial neural networks (ANN), while data transmission with low size is important in the field of source coding. In this paper, we constructed an alphabet of 36 letters which are encoded with the Huffman algorithm and then classified with a back-propagation Feed Forward artificial neural network. Since an ANN is initialized with random weights, the performance is not always optimal. Therefore, we designed a simple genetic algorithm (SGA) that choses an ANN and optimizes its architecture to improve the recognition accuracy. The performance evaluation is given to show the effectiveness of the procedure used, where we reached an accuracy of 100%.

Keywords: Character Classification, Huffman Codewords, Artificial Neural Networks, Genetic Algorithms, Neural network architecture.

1 Introduction

Artificial neural networks (ANN) are powerful data modeling tools that are able to capture and represent complex input/output relationships. The motivation for the development of neural network technology stemmed from the willing to develop artificial systems that could perform intelligent tasks similar to those performed by the human brain. According to [1], neural networks resemble the human brain since they acquire knowledge through learning. The acquired knowledge is then stored within inter-neuron connection strengths. The advantage of neural networks lies in their ability to represent both linear and non-linear relationships directly from the data being modeled.

The most common neural network models are known as *supervised learning* ANN and require a target vector in order to learn the input patterns. The goal of this network type is to create a model that maps the input pattern to the output by using historical data so that the model can then be used to produce the output when the desired input is unknown. Neural networks with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. Therefore, a trained neural network can be thought as an *expert* in the category of information it has been given to analyze. Other advantages include adaptive learning, self-organization, real time operation, fault tolerance via redundant information coding, etc. For short, the term artificial neural networks encompass a great variety of different software packages with many different types of artificial neurons, network architectures, and learning rules [2].

Many research efforts have been performed in pattern recognition fields [3,4,7]. Many of these approaches are based on having all the learning patterns before solving the classification problem, to design the algorithm.

One of the most classical applications in the ANN field is *character recognition* [8] with many applications in areas like; business, post office, bank, security system, etc. Furthermore, many other important fields are developed where character recognition can be applied, like optical character recognition and handwriting recognition. Indeed, a machine that reads banking checks can process many more checks than the human eye in the same time. This kind of application saves time and money, and eliminates the human factor, which would have to perform such a repetitive task.

Before starting the neural network training, we have to pay particular attention to the preprocessing stage of the input data. Firstly, patterns presented to the neural network need to have the same length. When constructing codewords with the Huffman algorithm, the resulting symbols have variable length size. We need to perform an strategy that builds all the input vectors to the same length size while keeping all the relevant information stored at the same time.

Neural networks sometimes tend to fall into *local optimum.* A local optimum of a combinatorial optimization problem is a solution that is either maximal or minimal, within a neighboring set of solutions. A global optimum is the optimal solution among all possible solutions. This problem can be reduced (but not avoided) by using a simple genetic algorithm that can optimize a given neural network's architecture by heuristically finding the best setting when choosing the optimal number of hidden layers and its neurons, activation functions, and training methods.

This paper is organized as follows; Section 2 describes the Huffman codewords construction while Section 3 presents the neural network design for the classification of the 36 characters in the alphabet constructed for the data transmission. In Section 5, the performance evaluation is given and analyzed. In Section 6 we explain the design and implementation of a simple genetic algorithm, and we show the results and their analysis in Section 6.1. The concluding remarks and future trends are shown in Section 7.

2 Huffman Codewords Construction

In 1951, David Huffman [9] gave the choice of a final exam and he had the idea of using a frequency-sorted binary tree to quickly prove this method to be the most efficient. Huffman codes are widely used in the area of compression and telecommunications given that the symbols have distinct probabilities of incidence. This property is used to advantage by tailoring the code lengths corresponding to those symbols in accordance with their respective probabilities of occurrence. Symbols with higher probabilities of incidence are coded with a shorter codeword, while symbols with lower probabilities are coded with longer codewords. However, longer codewords still show up, but tend to be less frequent and hence the overall code length of all codewords in a typical bit string tends to be smaller due to the Huffman coding.

Table 1 presents each of the 36 symbols in the alphabet we used with their random probability values. Table 2 shows the Huffman codeword of each symbol.

Table 1. A sample of generated random values for probability assignment to symbols

Symbols	0	1	2	3	4	5	6	7	8
Frequency	0.0469	0.0466	0.0464	0.0463	0.0463	0.0451	0.0442	0.0441	0.0438
Symbols	9	A	B	C	D	E	F	G	H
Frequency	0.0410	0.0398	0.0394	0.0387	0.0383	0.0366	0.0359	0.0341	0.0328
Symbols	I	J	K	L	M	N	O	P	Q
Frequency	0.0317	0.0317	0.0306	0.0264	0.0235	0.0204	0.0190	0.0135	0.0134
Symbols	R	S	T	U	V	W	X	Y	Z
Frequency	0.0083	0.0076	0.0069	0.0061	0.0047	0.0047	0.0022	0.0020	0.0017

Table 2. Source symbols with their corresponding codewords

Symbols	Codewords	Symbols	Codewords	Symbols	Codewords	Symbols	Codewords
0	1010	**9**	00100	**I**	01110	**R**	0100111
1	1011	**A**	00101	**J**	01111	**S**	1000100
2	1100	**B**	00110	**K**	10000	**T**	1000101
3	1101	**C**	00111	**L**	10010	**U**	1001110
4	1110	**D**	01000	**M**	000010	**V**	01001100
5	1111	**E**	01010	**N**	000011	**W**	01001101
6	00000	**F**	01011	**O**	010010	**X**	10011111
7	00010	**G**	01100	**P**	100011	**Y**	00111100
8	00011	**H**	01101	**Q**	100110	**Z**	100111101

3 Artificial Neural Network Design

3.1 Input Vectors - Target Vectors

The use of a neural network requires more attention on the data to be processed. Neural network training can be made more efficient if certain preprocessing steps are performed on the network inputs and targets. Since the input data are the codewords of the Huffman algorithm, an special care is required. Indeed, Huffman coding is a compression method that converts characters into variables length bit string [10]. Network-input processing functions transform inputs into a better form for the network use. Processing functions associated with a network output transform targets into a better form for network training, and reverse transformed outputs back to the characteristics of the original target data. In our case, since the Huffman codewords do not have the same length, we completed the stream of bits with a stream of 2 by preceding the codeword in order that each input data has a length of 9 characters, which is the value of the longest length in the Huffman codes alphabet constructed. Table 3 presents the codewords with their corresponding input vectors.

This strategy is efficient and can even help detect an error in a sequence of input vectors in a stream of bits obtained by the Huffman coding.

Each target vector has 36 components composed of 1' and 0's defined in the following way: the neural network should respond with a 1 in the position of the letter being presented to the network. All other values in the output vector should be 0.

Table 3. Source symbols with their corresponding codewords

Code-	Input	Code-	Input	Code-	Input	Code–	Input
words	Vectors	words	Vectors	words	Vectors	words	Vectors
1010	222221010	00100	222200100	01110	222201110	0100111	220100111
1011	222221011	00101	222200101	01111	222201111	1000100	221000100
1100	222221100	00110	222200110	10000	222210000	1000101	221000101
1101	222221101	00111	222200111	10010	222210010	1001110	201001110
1110	222221110	01000	222201000	000010	222000010	01001100	201001100
1111	222221111	01010	222201010	000011	222000011	01001101	201001101
00000	222200000	01011	222201011	010010	222010010	10011111	210011111
00010	222200010	01100	222201100	100011	222100011	00111100	200111100
00011	222200011	01101	222201101	100110	222100110	100111101	100111101

3.2 Neural Network Architecture

The network receives an input vector of nine features. It is then required to identify the letter by responding with an output vector of 36 elements. Each of the 36 elements of the output vector represents a letter.

The neural network's architecture is as follows:

– An input layer with 9 neurons (the total number of input features).
– An optional number of hidden layers (to be determined Heuristically).
– An output layer with 36 neurons (one neuron for each symbol in the alphabet).

The output is then passed through the competitive transfer function so that only one of the 36 outputs (representing the symbols in the alphabet), has a value of 1's. Competitive transfer function is a neural network transfer function that can calculate a layer's output from the network's input [5].

3.3 Learning Strategies

Learning (or training) in neural network theory means the adjustment of weights for the connections such that a cost function is minimized [11]. The neural network is first trained on ideal vectors until it has a low sum-squared error. Then, the network is trained on all sets of ideal and noisy vectors. The neural network is trained on 20 copies of noise-free alphabet at the same time as it is trained on noisy vectors. The 20 copies of the noise-free are used to maintain the network's ability to classify ideal input vectors. The training is done using the Levenberg-Marquardt algorithm [4], an algorithm implemented in [5].

The network is initially trained without noise. Data are divided into training, validation and test sets since there is only one sample of each symbol and we need to train on all of them. Validation data are used to stop training at the point where the network has generalized as well as it can and further training will only optimize the network's performance on the training set. The expense of generalization is measured by its performance on the validation set.

In addition, the neural network should be able to handle noise. The network should make as few mistakes as possible when classifying vectors with noise of mean 0 and standard deviation of 0.2.

4 Genetic Algorithms

Genetic algorithms [6] are proof implemented in a computer simulation in which a population of abstract representations (called chromosomes or the genotype of the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in a binary mode as strings of 0's and 1's, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens trough generations. In each generation, the fitness of every individual in the population is evaluated, while multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. This latter population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

5 Performance Evaluation

Early results obtained with the ANN having 20 neurons in the hidden layer are given in Table 4, where the average result from ten experiments is 82.78%.

Table 4. Recognition performance using the neural network classifier, from one out of ten experiments with a 69.44% accuracy

Alphabet symbols	Code– words	Recognition Performance	Alphabet symbols	Code– words	Recognition Performance
0	1010	100%	I	01110	100%
1	1011	100%	J	01111	100%
2	1100	100%	K	10000	100%
3	1101	100%	L	10010	0%
4	1110	0%	M	000010	0%
5	1111	100%	N	000011	0%
6	00000	0%	O	010010	0%
7	00010	0%	P	100011	100%
8	00011	100%	Q	100110	0%
9	00100	100%	R	0100111	100%
A	00101	100%	S	1000100	100%
B	00110	100%	T	1000101	0%
C	00111	100%	U	1001110	100%
D	01000	100%	V	01001100	100%
E	01010	100%	W	01001101	100%
F	01011	100%	X	10011111	100%
G	01100	0%	Y	00111100	0%
H	01101	100%	Z	100111101	100%

The Feed Forward (FF) ANN trained with the 36 symbols had an accuracy of 100% recognition performance in only six out of ten experiments, in the other cases the experiments had recognition performaces from 38.89% to 69.88%. Another Feed Forward Time Delay (TDFF) ANN was used to also perform ten different experiments, with the same architecture than the FF ANN. The average result from those experiments is 84.72%, with 100% accuracy in only three experiments, and had performances from 16.67% to 97.22% recognition accuracy. To asure that the ANN's will have an optimal accuracy, we need to perform some *fine tunning* on the network; heuristically search for the best neural network, number of hidden layers in the architecture, the number of neurons in each hidden layer, activation functions, and training method. Instead of performing trial and error experiments, we designed a SGA that optimizes these variables, and assures an accuracy of 100% in (almost) all the performed experiments; the GA design and implementation is explained in Section 6.

6 Simple Genetic Algorithm Design and Implementation

A SGA, was designed to improve the recognition accuracy of an ANN by searching for the optimal architecture configuration, this algorithm was benchmarked with the iris database [12], and then tested with the Huffman codewords. The SGA optimizes the pattern recognizer by firstly choosing an ANN and the best architecture. The SGA starts by randomly initializing a n number of binary chromosomes stated by the user. These individuals have a size of nine bits where each one representing a parameter to be optimized; the description of one individual is shown in Figure 1. Each individual represents an experiment to be performed; it chooses an ANN, a training method, a number of hidden layers, an activation function for each layer, and number of neurons on each layer. When the first experiments of a given generation are completed, we obtain a recognition performance ranging from 0 to 100%, which will be used as the fitness value for each genotype. Once each individual has obtained a fitness value, the rulette algorithm is performed to choose the next population of individuals, then all individuals go thorugh a crossover and mutation operations before performing a new set of experiments. The parameters to be optimized are described below.

– Type of ANN: two different ANN where used for these set of experiments, namely a Feed Forward and a Feed Forward Time Delay NN. More neural networks can be added at this point.
– Training Method: Two different training methods where used; scaled conjugate gradient back-propagation and Gradient descent with adaptive learning back-propagation, we can also consider more training methods.
– The number of hidden layers: right now the algorithm chooses from cero to four hidden layers even though more layers can be set.
– The activation function: two activation functions can be chosen, namely hyperbolic tangent sigmoid transfer function and the logarithmic sigmoid transfer function.
– The number of neurons: the number of neurons in each layer can be set from 1 neurons to 16, but more neurons can be stated by the user.

The SGA has a feature that makes the algorithm perform faster; each experiment proposed by a genotype is stored in a *dictionary*, and each time a new experiment is proposed, the algorithm searches through the dictionary, if the experiment has been performed before, it skips that experiment and goes to the next individual. With this method, the genetic algorithm performs faster since not all experiments

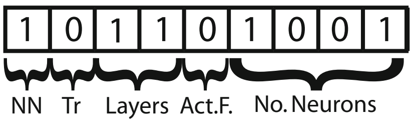


Fig. 1. Genotype design. The size can be dynamically modified if each variable needs more choices.

are executed in some generations. Of course the dictionary is empty every time the first generation of the SGA starts.

The SGA was tested with the *iris flower* database which has on three classes: Iris setosa, Iris virginica and Iris versicolor. each class has 50 samples where each sample holds four features. To perform our experiment, each class in the database was randomly divided into training and testing sets, 30 samples from each class to train the ANNs and 20 samples to test them.

6.1 Results and Analysis

With a population of 50 individuals in 50 generations, a crossover rate of $R_c = 0.5$ and a mutation rate of $R_m = 0.067$ the SGA along with an optimized ANN obtained a recognition performance of 98.33%, the genotype that achieved this accuracy is shown in Figure 2 and its fenotype is as follows,

- a Feed Forward Neural Network,
- trained with Scaled conjugate gradient backpropagation,
- one hidden layer,
- having 16 neurons, and
- a logarithmic sigmoid transfer function.

Once the SGA was tested with the iris database, we experimented with the Huffman codewords, using the same number of individuals, generations, and crossover and mutation rates, achieving an 100% recognition performance, as shown in Table 5. The winning genotype is shown in Figure 3 and its fenotype is as follows;

- a Feed Forward Time Delay Neural Network,
- trained with Scaled conjugate gradient backpropagation,
- two hidden layers,
- having 16 neurons, and
- a logarithmic sigmoid transfer function.

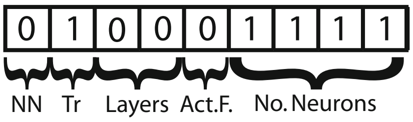


Fig. 2. Genotype that achieved the best accuracy with the iris flower database

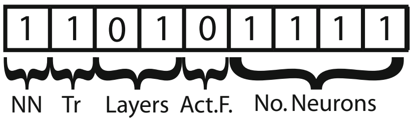


Fig. 3. Genotype that achieved the best accuracy with the Huffman codewords

Table 5. Recognition performance using the hybrid system

Alphabet symbols	Code–words	Recognition Performance	Alphabet symbols	Code–words	Recognition Performance
0	1010	100%	**I**	01110	100%
1	1011	100%	**J**	01111	100%
2	1100	100%	**K**	10000	100%
3	1101	100%	**L**	10010	100%
4	1110	100%	**M**	000010	100%
5	1111	100%	**N**	000011	100%
6	00000	100%	**O**	010010	100%
7	00010	100%	**P**	100011	100%
8	00011	100%	**Q**	100110	100%
9	00100	100%	**R**	0100111	100%
A	00101	100%	**S**	1000100	100%
B	00110	100%	**T**	1000101	100%
C	00111	100%	**U**	1001110	100%
D	01000	100%	**V**	01001100	100%
E	01010	100%	**W**	01001101	100%
F	01011	100%	**X**	10011111	100%
G	01100	100%	**Y**	00111100	100%
H	01101	100%	**Z**	100111101	100%

7 Concluding Remarks and Future Trends

The performance of an ANN on its own sometimes gives an 100% accuracy, but the architecture always has to be tuned heuristically by performig *trial and error* experiments. Even though the patterns are well defined, as shown in the Huffman codewords, an ANN can sometimes reach a local optimum and a recognition performance as low as 16.67%. The SGA simplifies the tuning task and avoids performing low performance experiments. Furthermore, the *dictionary* feature in the SGA helps us perform the necesary experiments, and avoids repeating previus ones. As we can see on the fenotypes from both experiments, with the iris database and the Huffman codewords, the SGA always chooses the highest number of neurons in each hidden layer, nevertheless, two hidden layers are enough to achieve a good performance.

On future works, we are planing to apply this SGA on other pattern recognition tasks taking into account more configurations; we don't think more hidden layers are necessary to this point, given that more hidden layers means more training time. Another topic is about the designing of a neural network architecture with variable length of the input vector for codewords discrimination.

References

1. José, C.P., Neil, R.E., Curt, W.L.: Neural and Adaptive Systems: Fundamentals through Simulation, ISBN 0-471-35167-9
2. Dong, X.N.: Application of Neural Networks to Character Recognition. In: Proceedings of Faculty Research Day, CSIS, Pace University (May 2007)

3. Motameni, M.A., et al.: Learning Flexible Neural Networks for Pattern Recognition. In: Proceedings of World Academy of Science, Engineering and Technology, vol. 21 (2007) ISSN 1307 6884
4. Bishop, C.: Neural Networks for Pattern Recognition. Oxford Press, Oxford (1995)
5. Neural Network Toolbox User's Guide, `www.mathworks/access/helpdesk/help/pdf_doc/nnet/nnet.pdf`
6. Fogel, D.B.: Evolutionary Computation: Towards a New Philosophy of Machine Intelligence, p. 140. IEEE Press, New York (2000)
7. Lezray, O., Fournier, D., Cardot, H.: Neural Network Induction Graph for Pattern Recognition. Proceedings of Neurocomputing 57, 257–274 (2004)
8. Araokar S.: Visual Character Recognition using Artificial Neural Networks (2010), `http://www` (visited on April 5-10, 2010)
9. Huffman, D.: Profile Background Story. Scientific American, 54–58 (September 1991)
10. Hamid, S., Ismail, A., Siddiqui, A.A.: Practical workbook, Information Theory. Department of Computer and Information Systems Engineering, NED University of Engineering and Technology, Karachi - 75270, Pakistan (2010)
11. `http://en.wikipedia.org/wiki/Neural_network` (visited on March 25, 2010)
12. Frank, A., Asuncion, A.: (UCI) Machine Learning Repository University of California, Irvine, School of Information and Computer Sciences (2010), `http://archive.ics.uci.edu/ml`

Fraud Detection Model Based on the Discovery Symbolic Classification Rules Extracted from a Neural Network

Wilfredy Santamaría Ruiz and Elizabeth León Guzman

Univerdidad Nacional de Colombia - Bogotá Colombia - 2010
{wsantamariar,eleonguz}@unal.edu.co

Abstract. This paper presents a fraud detection model using data mining techniques such as neural networks and symbolic extraction of classification rules from trained neural network. The neural network is first trained to achieve an accuracy rate, the activation of the values in the hidden layers of the neural network is analyzed and from this analysis are generated classification rules. The proposed approach was tested on a set of data from a Colombian organization for the sending and payment of remittances, in order to identify patterns associated with fraud detection. Similarly the results of the techniques used in the model were compared with other mining techniques such as Decision Trees and Naive Bayes. A prototype software was developed to test the model, which was integrated into RapidMiner tool, which can be used as a tool for academic software.

Keywords: Data mining, neural networks, rule extraction, detection of fraud, experts.

1 Introduction

Nowadays many companies globally, use data mining techniques to recognize patterns of fraudulent transactions. Among the techniques that have improved performance classification according to [1], [2], [3] are the neural networks given their high level of accuracy in the prediction and robustness to noise in the data. However the neural networks have some disadvantages such as the learning time is high and the results are not explained is a black box, in many applications is highly desirable to extract the classification rules, so that experts easily interpret and verify the results. In recent years have developed algorithms that allow extraction of rules from the neural network to understand the classification results generated by the network, techniques such as clustering and decision trees [4],[5],[6] have been used with good acceptance. These techniques have been experimentally tested on problems of classification in synthetic data bases such as Iris, MONK's Problems, prognosis of hepatitis and heart disease.

This paper presents a case of using neural networks and symbolic rules extraction in a Colombian organization. Specifically, we analyze the process of sending and payment of remittances, as well as the buying and selling of foreign currency in order to identify patterns related to money laundering. Initially, there are activities that involve pre-processing to obtain high quality data, activities such as elimination of duplicate records, elimination of attributes that does not contribute with information, detection of missing values and data inconsistency were made to preprocess the data. Finally, we apply a neural network model in combination with an algorithm for extracting symbolic rules, which was applied to detect possible money laundering. The organization of this chapter is as follows: Section 2 describes the topic of extracting symbolic rules from a neural network in Section 3 provides a description of the proposed detection model, Section 4 presents the results of applying the proposed detection model, consisting of a neural network and symbolic extraction of rules, as well as comparison with other mining techniques such as Decision Trees and Naive Bayes, Section 5 presents the conclusions and future work, and Section 6 references.

2 Extraction of Rules from a Neural Network

In the last two decades have seen a lot of research and applications of neural networks for a variety of classification problems in the real world. Several empirical studies have indicated that there are some problem domains for which neural networks offer superior classification accuracy compared to other learning algorithms [7].

One of the major disadvantages of neural networks is the lack of transparency in the knowledge generated, given the structure and parameters of the connections between network elements. That's why, that at the beginning of the nineties began to receive the concept of rule extraction (RE) to express the knowledge embedded in a neural network. Various concepts and algorithms have been developed to date [8] gives a comprehensive collection of research in this field.

There are two main concepts in Rule Extraction. Both have in common that they take a trained ANN. The ANN (or its weights) is analyzed to extract a whole rule-set which should represent the knowledge embedded in the ANN [9]. In Figure 1 this simple context is shown. The two main concepts are the following:

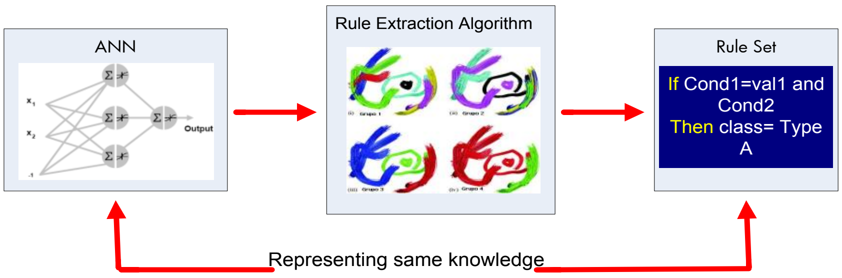

ANN
x1
x2
-1
Output
Rule Extraction Algorithm
Rule Set
If Cond1=val1 and Cond2 Then class= Type A
Representing same knowledge

Fig. 1. Principle on rule extraction from trained ANN

i) Black-Box. We can treat an ANN as a black box, without considering its architecture and weight values. Input vectors are mapped by the ANN to output vectors and the RE-algorithm is extraction rules by analysing the relation between in- and output. In some literature algorithms of this kind are also called “pedagogical” and **ii)** Decompositional. This type of algorithm takes a look at the weights itself. There are algorithms which try to cluster similar weights to simplify rules. Others search for dominant paths through the net.

2.1 Rule Extraction Algorithms

The main algorithms developed in the extraction of the rules of a neural network according to Andrew [8] and that have been implemented in applications real are:

1. **MofN Algorithm.** It was published by Towell & Shavlik [10], is based on the algorithm "SUBSET". It differs from this in the way of doing the search for the rules of the form: IF (M of the following N antecedents are true) THEN. The steps of the algorithm are: **i)** With each hidden and output unit, form groups of similarly weighted links, **ii)** Set link weights of all group members to the average of the group, iii) Eliminate any groups that do not significantly affect whether the unit will be active or inactive, **iv**) Holding all link weights constant, optimize biases of all hidden and output units using the backpropagation algorithm, **v**) form a single rule for each hidden and output unit. The rule consists of a threshold given by the bias and weighted antecedents specified by the remaining links and **vi**) Where possible, simplify rules to eliminate superfluous weights and thresholds.
2. **VIA algorithm.** The Validity Interval Analysis (VIA, VI-Analysis) was invented by Thrun in [11]. VIA is a “generic tool for analyzing Backpropagation-style artificial neural networks”. It’s a method for proving rules by propagating whole intervals through the net and making statements about the range where in- and outputs can lie. The rules extracted by VIA are if-then rules. There are two possible outcomes of the analysis process "VIA” : **i**) the routine converges. The resulting intervals are a sub-space that includes all the activation patterns consistent with the initial schedule or **ii**) Conflict (i.e. an empty interval)**.** If you build an interval is empty then the lower limit of the range exceeds its upper limit, thus there will be some activation pattern that satisfies the constraints imposed by the initial interval of validity. Therefore, the initial intervals are incompatible with the weights and biases of the network.
3. **SUBSET Algorithm.** It was developed by Towell [10] is one of the first approaches to decompositional algorithm. "SUBSET" is a search based on the extraction of rules of the form IF - THEN from a MultiLayerPerceptron network (SPLM) binary output units. Additionally, the network has only binary input values. The algorithm finds the combination of positive weights connected to a node that is active. These sets of connections with negative weights are combined to form rules that activate the node. The algorithm performs an exhaustive search on all incoming connections from one node and, as such, it is necessary to restrict the search space.
4. **RULENET algorithm.** The RuleNet technique and the connectionist scientist game of McMillan [12], is one of the earliest examples in which a specialised ANN training regime incorporating the decompositional approach is used as the basis for extracting Boolean rules. The basic modus operandi in RuleNet is to iterate through the following steps: i) train an ANN, ii) extract the symbolic rules (using the connection strengths in the network), and iii) inject the extracted rules

back into the network. The process terminates when the resulting base of extracted rules adequately characterises the problem domain.

3 Fraud Detection Model Based on Neural Networks and M of N Algorithm

The problem of fraud has represented losses throughout history. Most financial institutions worldwide have adopted strategies to reduce the risk of fraud, strategies among which are the use of modern detection systems to determine, through a probabilistic rating or application of rules, when a transaction is fraudulent. Many of these systems implement classification and prediction models through data mining techniques such as neural networks, which is achieved with high accuracy [1], [2] but its major criticisms are: the time learning is expensive and way of classification, which is a black box, and in many applications is highly desirable to extract the classification rules, so that experts easily interpret and verify the results.

This paper presents a fraud detection model, based on data analysis using data mining techniques, specifically considering the use of the technique of extracting symbolic classification rules from a neural network trained "MultiLayerPerceptron" for characterization of patterns of fraud, in order to assist the expert of the business to more easily examine and verify the results obtained to support decision making. The proposed model is based on a binary classification of usual and unusual transactions as illustrated in figure 2, which presents an overview of the interaction of different modules and communicating with elements outside the system (databases and files).

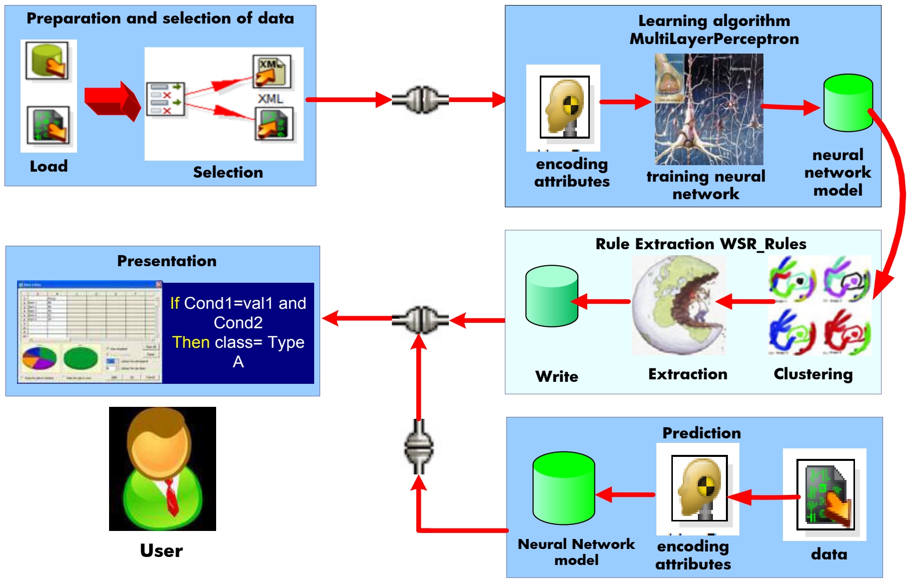


Fig. 2. Model proposed fraud detection

The model consists of several modules that provide specific functionality in order to obtain the hidden knowledge that exists in the dataset. The modules are:

1. **Preparation and selection of data.** Responsible for the connection to the source, the selection and construction of the dataset for the extraction of knowledge.
2. **Extraction of knowledge.** It is the engine model proposed detection, consisting of:
   i) Learning algorithm. The process builds and trains a neural network "Multi-layerPerceptron". The network model is built on based to the number of attributes and classes in the training dataset envelope which applies the "backpropagation" algorithm to obtain the trained neural network model that is validates against the entire data.
   ii) Adaptation of the rule extraction algorithm. Among the great variety of algorithms and extraction methods carried out to date, work in this article focused on the MofN algorithm proposed by Towell [10], this algorithm was adapted as it describes in figure 3. The rules extracted are the shape: IF (M of the following N antecedents are true) THEN.
3. **Presentation of results.** Responsible for the interaction with the user and the presentation of the final results of the learning process.
4. **Prediction.** Responsible for inferring a prediction on a set of new data. This is part found knowledge to the classification of the data set.

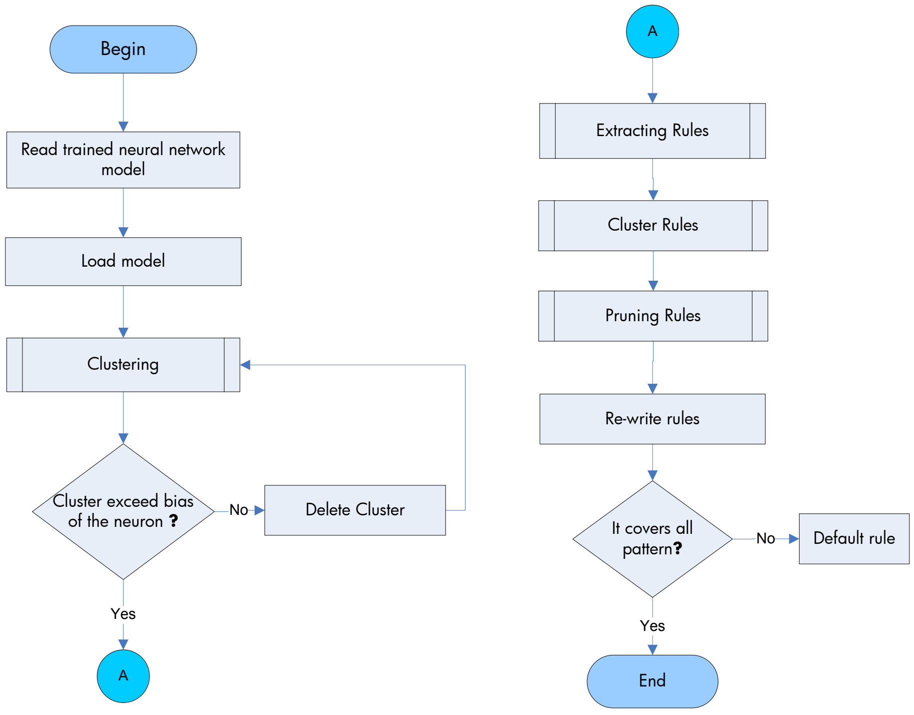


Fig. 3. Adaptation M of N Algorithm for the fraud detection model

4 Experimentation and Results

The proposed model was applied to a Colombian organization that for the purposes of this article call it "XYZ Limited". This organization is dedicated to providing services related to foreign exchange as well as sending and payments remittance.

The information includes data of successful transactions and denied this is stored in: i) Customers. Those making shipment, receipt, buy and sell currencies. The information is detailed in terms of basic personal data, financial information (economic activity, income, wealth) and additional information, ii) Transactions made by customers. The information is detailed on the amount of the transaction, type of operation, tax liens, collection site and details of the beneficiary and / or payer. With the present controls the transactions are classified as normal or unusual.

After preliminary analysis, it is obtained 14 attributes have a great utility in data mining; these ones are: product, region target, country source, sex, marital status, economic activity, use of resources, cumulative amount, age, monthly income, number of transactions, entailed days, transaction status and client status. Taking completely different operations with the selected final attributes, the set of records is reduced to 157200. More information selection process carried out in the article [13] presents a more detailed description.

4.1 Sampling

Given the number of data about the type of the class of transactions, 97.29% are labeled data as usual and only 2.71% to unusual, due to the limited number of examples of unusual class (E). We carry out a stratified sampling of proportional in order enhance the number of examples of unusual transactions and have a sample both classes. For doing use the functionality provided by Oracle named "ORA HASH" [14]. In this scenario, the unusual class with 3207 records is increased (75.35% of total unusual records) and 4190 records from the usual class (2.67% of total usual operations). This sample was used as a set of training and testing for the evaluated different techniques.

4.2 Parameters Setting

The algorithms the neural network, decision trees (J48) and Naive Bayes were implemented with the tool RapidMiner [15]. The classification models used, used the following configuration:

- Cross-validation 10 folds.
- Validation data set: 150.00 records.
- Training and Test data set: 7397 records.
- For the neural network are optimized the following parameters: **i**) Number of hidden layers and neurons = one hidden layer and eight neurons, **ii)** Momentum= 0.0029999999, **iii**) Rate learning = 0.4 and **iv)** the sigmoid function was used as

activation function for nodes in the hidden layer and output. The number of iterations is defined by 500 times.

- For the decision tree was optimized the following parameter: **i)** confidence Factor = 0.25 used for pruning and **ii)** The minimum number of instances per leaf = 2

4.3 Results and Comparison Model Classification

In the table 1 is shows the results obtained by the neural network, J48 y Naive Bayes in term confusion matrix. The table 2 shows the results taking into account the labeled as unusual class (E), which are the main interest of this investigation, for the following measures: Accuracy, Recall, Error, False Positive Rate (FP rate) and False Negative Rate (FN rate).

In the same way, it was performed a behavioral analysis through ROC curves over classes labeled as unusual (E). Figure 4 shows a comparison between Naïve Bayes, J48 and NN algorithms.

Analyzing the above results, the algorithm that gives better results for classifying classes labeled as unusual (*E*) is the neural network with 2% of precision in the classification that decision tree and 6% to Naïve Bayes. Similarly, reviewing the ROC curves and area under these, all algorithms show good performance with an area greater than 0.9, emphasizing the neural network with a value of 0.962. Nevertheless, results obtained by J48 are close enough to NN. On the other hand, Naïve Bayes has a poor performance and is no convenient for this kind of problem, especially for the finding of unusual transactions; this judgment can be re-evaluated in a more complex analysis.

Table 1. Confusion matrix for classification algorithms

	Prediction					
	Neural Network		J48		Naïve Bayes	
	Usual	Unusual	Usual	Unusual	Usual	Unusual
Usual (N)	138219	14725	139412	13529	135336	17533
Unusual (E)	**289**	3967	**370**	3886	**548**	3708

Table 2. Comparison of classification algorithms over data

	Neural Network	Decision Tree(j48)	Bayes
Accuracy	90.45%	91.16%	88.49%
Precision	21.22%	23.31%	17.46%
Recall	93.21%	91.31%	87.12%
Error	9.55%	8.84%	11.51%
FP Rate	9.62%	8.84%	11.46%
FN Rate	6.79%	8.69%	12.87%

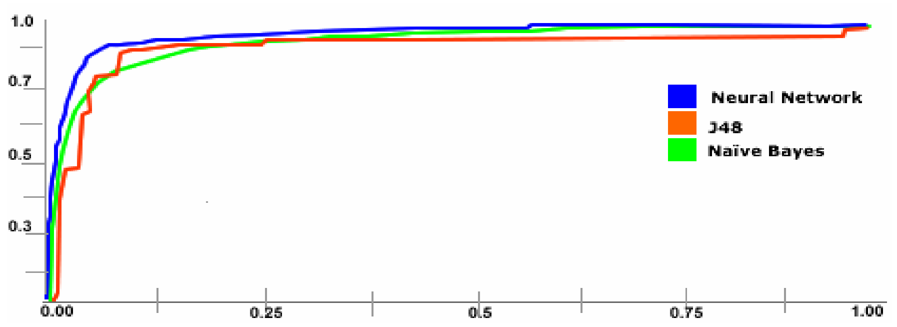


Fig. 4. ROC curves for the classification algorithms

Finally, the time used for training and validation set are shown in Table 3.

Table 3. Time set training and validation[1]

Algorithm	Training(Seg)	Validation(Seg)
Neural Network	6660	150
Decision Tree	15	5
Bayes	5	5

The previous table shows that the spent time by the neural network is notoriously high compared to the other models, which contrasts with its good performance of classification of false negatives, which are the focus of this work.

4.4 Hypothesis Testing

Although performance measures seems to be sufficient, it is necessary to prove if exist a significant difference between the shown models. For evaluating the classifier we used ANOVA test. In this case, we have a Null Hypothesis: $H_0 = \mu_1 = \mu_2 = \ldots \mu_L$ where μ_1 is the average error and L is the number of algorithms. Then we have L algorithms and K observations: $X_{i,j} \sim N(\mu_j, \sigma^2)$ j=1... L i=1... K

Finally, the according Alpaydin [17] is accepting the null hypothesis H_0 (all the means are equal) if given a value of significance; the expected error rate is less than $F\alpha_{, L-1, L(k-1)}$, in other case, H_0 is rejecting.

In this case, the focus is on testing whether the neural network model vs. Decision Tree and Naive Bayes significant differences in means error. For this task, we used the following parameters: Hypothesis: $H_0 = \mu_1 = \mu_2 = \ldots \mu_L$, with L=3, K=10 and Confidence (95%) = (α=0.05). The table 4 shows the results obtained. The probability obtained given a confidence of 95% is considerably less than the alpha value set ($Pr = 0.0000001198 < \alpha = 0.05$), indicating that the average errors between the three classifiers (NN, J48 and Bayes) is different and have a significant change and one of

[1] All the experiments were run on a computer with the following specs: RAM: 3 GB Intel Core 2 Duo T5550 / 1.83 GHz.

Table 4. Analysis of variances for NN and other models[2]

Variation	SS	DF	MSS	F(Exp)	F(Teo)	Prob
SSB	0.009	2	0.0045	60.810	3.3541	0.000000011
SSW	0.002	27	0.000074			
Total	0.011	29				

these algorithms have better performance than the others. Similarly, in accordance with the above analysis we can see that: $60.810 \neq F_{(2, 27)} < F_{(0.005, 2, 27)} = 3.3541$, and the null hypothesis is rejected.

Given the above results, in order to determine which algorithm is the best performance, a new comparison was made between the neural network model and decision trees, just that the neural network model and probabilistic model Naive Bayes with the following parameters: Hypothesis: $H_0 = \mu_1 = \mu_2 = \ldots \mu_L$, with L=2 , K=10 and Confidence (95%) = (=0.05).

In the table 5 shows the results of the comparison of the neural network vs. the decision tree where $3.245635 \approx F_{(1, 18)} < F_{(0.005, 1, 18)} = 4.4138$, indicating that the average errors between the two classifiers (NN and J48) are similar and no significant variation to determine which one is better than another.

Table 5. Analysis of variances for NN and J48

Variation	SS	DF	MSS	F(Exp)	F(Teo)	Prob
SSB	0.000157	1	0.000157	3.245	4.4138	0.10545
SSW	0.000871	18	0.000048			
Total	0.001028	19				

Similarly, in the table 6 shows the results of the comparison of the neural network vs. Naive Bayes. In this case, the average errors between the two classifiers (NN and Bayes) are different and have significant variation, so the neural network algorithm has better performance. In accordance with the above analysis we can see that: $180.139 \neq F_{(1, 18)} < F_{(0.005, 1, 18)} = 4.4138$, and the null hypothesis is rejected.

Table 6. Analysis of variances for NN and Bayes

Variation	SS	DF	MSS	F(Exp)	F(Teo)	Prob
SSB	0.00785	1	0.00785	180.139	4.4138	0.10545
SSW	0.00078	18	0.000043			
Total	0.00863	19				

[2] The results obtained the ANOVA tests are shown with the following information: SS= Square sum, MSS= Mean square sum, DF= Degrees of freedom, F (Exp) = F distribution experimental, F (Teo) = F distribution theoretical, Pro= Probability, SSB= Sum of squares between groups and SSW= Sum of squares of the Group.

4.5 Extraction of Rules - MofN Algorithm

The grouping distance parameter used in the algorithm was 0.25, according to the literature the most optimal [9]. During the initial phase extraction were obtained 140 rules, in terms of the relationship between the input layers and hidden, just as between the hidden layer and output. After the process of pruning and re-writing the rules in terms of inputs and outputs are finally 51 rules. The rules obtained were validated for the entire data set. Here are the most important rules:

- ✓ IF (1 of (Attrib->estadoCliente=I)) THEN E Confidence: 20%
- ✓ IF (1 of (Attrib->estadoCliente=E)) THEN E Confidence: 60%
- ✓ IF (1 of (Attrib->montoAcumulado=>=15981)) THEN E Confidence: 27%
- ✓ IF (2 of (Attrib->estadoCliente=I, Attrib->paisOrigen=JAPON)) THEN E Confidence: 100%
- ✓ IF (1 of (Attrib->estadoCliente=R)) THEN E Confidence: 21%
- ✓ IF (1 of (Attrib->estadoCliente=A)) THEN N Confidence: 98.35%
- ✓ IF (1 of (Attrib->paisOrigen=DESCONOCIDO)) THEN N Confidence: 100%
- ✓ IF (1 of (Attrib->estadoCivil=SOLTERO) THEN N Confidence: 97.04%
- ✓ IF (1 of (Attrib->montoAcumulado=<=769)) THE N Confidence: 98.57%
- ✓ IF (1 of (Attrib->montoAcumulado=769-1514)) THEN N Confidence: 98.7%
- ✓ IF (1 of (Attrib->montoAcumulado=1514-2259)) THEN N Confidence: 98.91%
- ✓ IF (1 of (Attrib->diasVinculacion=1160-1278)) THEN N Confidence: 97.99%
- ✓ IF(3 of (Attrib->actividadEconomica=HOGAR, Attrib->paisOrigen=ECUADOR, Attrib->numeroTransacciones=40-50, Attrib->numeroTransacciones=30-40)) THEN N Confidence: 95.23%
- ✓ IF (1 of (Attrib->actividadEconomica=EMPLEADO)) THEN N Confidence: 97.27%

Given the model were obtained rules that characterize unusual cases (27 rules) and usual cases (24 rules). For the usual cases the rules have a confidence greater than 97%, while for unusual cases, the average is less than 50%.

4.6 Association Rules

The JRIP algorithm was used. This technique obtained 16 rules for unusual cases with confidence in average is less than 50%. Examples of rules generated are:

- ✓ (estadoCliente = E) => txestado=E Confidence:60%
- ✓ (montoAcumulado = 5430-10661) and (numeroTransacciones = 1-10) => txestado=E Confidence: 24%
- ✓ (montoAcumulado = >=15981) => txestado=E Confidence:27%
- ✓ (montoAcumulado = 10661-15981) => txestado=E Confidence: 18.21%
- ✓ (montoAcumulado = 3749-5430) and (numeroTransacciones = 1-10) and (producto = SEND) => txestado=E Confidence: 14.98%
- ✓ (montoAcumulado = 3749-5430) and (actividadEconomica = ESTUDIANTE) => txestado=E Confidence: 15.70%
- ✓ (montoAcumulado = 5430-10661) and (departamentoDestino = VALLE-DEL-CAUCA) and numeroTransacciones = 10-20) => txestado=E Confidence:4.86%

4.7 Decision Tree

The J48 was used, which were obtained 377 rules, 84.88% of usual cases and 15.12% for unusual cases. Examples of rules generated are:

- ✓ IF estadoCliente = I: E Confidence: 20%
- ✓ IF estadoCliente = R: E Confidence: 21%

- ✓ IF estadoCliente = E: E Confidence: 60%
- ✓ IF estadoCliente = A| montoAcumulado = 10661-15981: E Confidence: 17.52%
- ✓ IF estadoCliente = A | montoAcumulado = 3749-5430 numeroTransacciones = 1- 10: E Confidence: 7.32%
- ✓ IF estadoCliente = A | montoAcumulado = 5430-10661 numeroTransacciones = 1-10: E
- ✓ Confidence: 23.60%
- ✓ IF estadoCliente = A | montoAcumulado = 2259-3749 | numeroTransacciones = 1-10 actividadEconomica = ESTUDIANTE: E Confidence: 14.39%
- ✓ IF estadoCliente = A | montoAcumulado <=769: N Confidence: 99.84%
- ✓ IF estadoCliente = A | montoAcumulado =1514-2259: N Confidence: 99.66%
- ✓ IF estadoCliente = A | montoAcumulado =5430-10661|numeroTransacciones = 20-30: N Confidence: 97.87%
- ✓ IF estadoCliente = A | montoAcumulado =5430-10661| numeroTransacciones = 10-20| paisOrigen = AFGANISTAN: N

Analyzing the rules obtained on the entire set of data, 81.16% have zero confidence, which does not add value.

4.8 Comparison Rules Extraction Models

The algorithm for extracting rules from the network reported unusual rules and usual rules, while the association rules algorithm had acted in breach, noting only unusual classes. Likewise, the decision tree was the only one present rules with a confidence equal to zero (81% of all generated rules.) It is highlighted that confidence to the usual rules more than 97%, however, the classification rules is unusual below 50%.

In the table 7 presents a comparative summary of the results of testing of the extraction algorithm implemented and. other techniques.

Table 7. Experimental results rule extraction techniques

Algorithm	#Rules	# Usual	#Unusual	Confidence=0	Time(seg)
WSR-M of N	51	24	27	0	12
J48	377	320	57	306	4
JRip	16	0	16	0	18

5 Conclusions

The neural networks are one of the approaches used for inductive learning and have demonstrated good predictive behavior in a variety of interesting problems (i.e. fraud detection). However they have a great limitation, its learned hypothesis is often incomprehensible. To deal with this limitation, a number of research groups have developed techniques for the extraction of rules, which is to represent the functions of a neural network in one language, as rules of inference symbolic, which facilitates a better understanding. The focus of this paper has been adapted a neural network model and a method for extraction of rules, called MofN as part of a model of detection of fraud, in order to help examine and verify the expert business more easily results in support of decision making.

The adaptation in this thesis, MofN algorithm has a clear benefit on other algorithms from extraction rules, because it requires no make pruning neural network to extract rules, which prevents a re-training, reducing processing times. This approach is scalable for large networks and problems with large spaces features domains.

The results of the model proposed on the validation set (all data) were excellent performance rating, with an accuracy of the 90.45% for cases usual and 93.75% in unusual cases.

The algorithm of extraction rules implemented in this paper applied on the trained neural network obtained rules of classification, which allows to know the attributes taken into account by the neural network to classify analyzed classes. The confidence of the rules generated for usual cases is greater than 97%, while for unusual cases confidence in its majority is less than 50%. This is due to the low number of records in unusual cases; however, the result is similar to other techniques as trees decision (J48) and Association (JRIP) rules.

The rules obtained from the algorithm implemented in this study vs. the rules generated from decision tree (J48) to the data set studied, the decision tree rules are so numerous that an expert is a high operating loads, while the rules obtained with the proposed algorithm are less 20% of the rules of the tree. Similarly we found in the different experiments that 80% of the decision tree rules have zero confidence.

The rule extraction processes are linear behavior, which contrasts with the learning process of neural networks, caused that time to extract the classification rules can not represent the entire model embedded in the network.

The ANOVA test, which is taken as performance measure " recall ", it was found that neural networks and decision trees do not show a statistically significant difference to say that an algorithm is better than the other, otherwise the model of Naive Bayes classification, the difference if significant and therefore it is best to use the neural network.

Future Works

The results presented in this paper, indicate that the extraction rule "M of N" method is able to extract from a trained neural network model rules without have to pruning of the network, which is a feature to highlight in this method, compared to other techniques used in the extraction rule. However, this method has some limitations suggested as a future studies:

- Clustering process. It is suggested to use different clustering techniques to build groups of similar weights to each neuron input of the network, in order to optimize the rules generated in the process.
- Optimal distance grouping. It is suggested to determine the current clustering process, the optimal value of distance required for the formation of groups.
- Independence of the neural network architecture. We recommend further studies for the combination of techniques for extracting rules, independent of the neural network architecture employed.
- Presentation of the rules learned by "M of N". Although the rules extracted from neural network by "M of N" are quite understandable, is to explore alternative representation.

References

1. Kou, Y., Lu, C.T., Sirwongwattana, S., Huang, Y.P.: Survey of fraud detection techniques. In: Proceedings of IEEE Intl Conference on Networking, Sensing and Control (2004)
2. Shen, A., Tong, R., Deng, Y.: Application of classification models on credit card fraud detection. In: International Conference on Service Systems and Service Management, pp. 1–4 (2007)
3. Shao, H., Zhao, H., Chang, G.: Applying data mining to detect fraud behavior in customs declaration, vol. 3 (2002)
4. Lu, H., Setiono, R., Liu, H.: Effective data mining using neural networks. IEEE Transactions on Knowledge and Data Engineering 8(6), 957–961 (1996)
5. Setiono, R., Leow, W.K.: Fernn: An algorithm for fast extraction of rules from neural networks. Applied Intelligence 12(1), 15–25 (2000)
6. Fu, L.: Rule generation from neural networks. IEEE transactions on systems, man and cybernetics 24(8), 1114–1124 (1994)
7. Craven, M.W.: Extracting comprehensible models from trained neural networks. PhD thesis, University of Wisconsin -Madison (1996)
8. Andrews, R., Diederich, J., Tickle, A.B.: Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowledge-Based Systems 8(6), 373–389 (1995)
9. Mayer, H., Huber, Rohde, Tamme: Rule extraction from artificial neural networks. University Salzburg (October 12, 2006)
10. Towell, G., Shavlik, J., Noordewier, M.: Refinement of approximate domain theories by knowledge-based neural networks, pp. 861–866 (1990)
11. Thrun, S.B.: Extracting symbolic knowledge from artificial neural networks. Revised Version of Technical Research Report I-University Bonn (1994)
12. Mcmillan, C., Mozer, M.C., Smolensky, P.: Rule induction through integrated symbolic and subsymbolic processing. In: Advances in Neural Information Processing Systems, pp. 969–976. Morgan Kaufmann, San Francisco (1992)
13. Andrés Ramírez, J.G.E.L., Santamaría, W.: Análisis del proceso de giros internacionales y control de lavado de activos mediante minería de datos. Revista Tendencias en Ingeniería de Software e Inteligencia Artificial 2 (2008)
14. Oracle, Database sql reference-ora hash (2003), Online `http://download.oracle.com/docs`
15. Rapidminer.: Rapidminer- free library to work data mining (2002)
16. Quinlan, J.R.: Induction of decision trees. Machine learning 1(1), 81–106 (1986)
17. Alpaydin, E.: Introduction to Machine Learning. The MIT press, Cambridge (2004)

Hardware Implementation of Artificial Neural Networks for Arbitrary Boolean Functions with Generalised Threshold Gate Circuits

Maciej Nikodem

Wrocław University of Technology
Institute of Computers, Control and Robotics
ul. Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
maciej.nikodem@pwr.wroc.pl

Abstract. This paper describes nanocircuits that draw on negative differential resistance and are capable of implementing complex threshold functions in a single gate structure. Due to nanometer dimensions, high operational frequencies and low power consumption these devices can be used for efficient hardware realisation of artificial neural networks (ANNs). We present state of the art in development of such circuit and focus on Generalised Threshold Gates (GTGs) that are capable of implementing arbitrary Boolean functions in a single gate structure. Algorithm for implementing Boolean functions outputs circuits with predefined weights and thresholds. This enables to construct application specific ANNs and eliminates the requirement for network learning when this kind of gates are used.

Keywords: Threshold logic, hardware implementation, negative differential resistance, synthesis.

1 Introduction and Previous Work

Threshold functions and artificial neural networks (ANNs) are known for many years and have been thoroughly analysed. However, most works focused on theoretical capabilities of such elements and their software implementation rather then constructing such elements in hardware. Not much attention was also paid to ANN implementing Boolean functions, which, in our opinion, is due to two main reasons:

– if we assume the general model of ANN's neuron then it can be easily shown, that all Boolean functions of n variables can be implemented with 2 level neural networks,
– for more then 50 years it was uneconomical and infeasible to construct threshold gates due to technology difficulties.

Despite technology obstacles several authors [1,2,3,4,5] studied theoretical possibilities of implementing Boolean functions using threshold logic. This was motivated by the fact that threshold gates, when implemented, would simplify the implementation of such functions. Theory of threshold elements and ANNs was developed without a practical hardware realisation of such elements [5,6,7,8]. Comprehensive work by Muroga [4] and Kohavi [5] defined parameters and prove properties of switching circuits and threshold networks. Their result were later used in the theory of artificial neural networks to determine their capabilities. In particular it was presented that network build of linear perceptrons can implement only linearly separable functions, that constitute a subset of all Boolean functions of n variables. This motivated others [6,8] to search for different types of neurons. As a result neurons with square and polynomial activation functions were proposed and proved that they enable to implement larger number of different Boolean functions. This results where then used to analyse different aspects of threshold networks and their capabilities. Unfortunately, majority of works have focused on simple threshold gates, synthesising threshold networks and algorithms for representing given Boolean function as a minimal sum of simple threshold functions [4,5,9].

Above mentioned results were found interesting in recent years as technology improved and enabled to construct devices that implement threshold functions. Using threshold logic to implement Boolean functions became even more interesting as these devices can be developed in nanotechnology. Small dimensions and drawing on phenomenon that are parasitic for traditional CMOS technology, e.g. Coulomb blockades, electron tunnelling and so on, are another advantages of such circuits. Advances in nanotechnology enabled to construct electronic devices that are feasible and immanently implement threshold functions. This extends practical applications of threshold theory and enables to construct networks of threshold elements that implement Boolean functions. On the other hand, recent advances in nanodevices have showed that NDR-based gates can implement also more complex threshold functions. Therefore, single nanogates may implement larger set of complex Boolean functions and lead to further simplification in threshold networks [10,11]. Above mentioned capabilities of NDR-based devices has been widely known for last few years, however scientists have focused on traditional threshold logic and linear neural network applications. Less attention was paid to threshold gate synthesis and there is no single paper on how NDR gates affect real life application of artificial neural networks for Boolean function synthesis.

The following sections will briefly introduce NDR-based circuits, their limits and ability to implement different Boolean functions. Later on we will give a formal prove that NDR gates can be used to construct ANNs that are capable of implementing any Boolean function.

2 Hardware for Implementing Threshold Elements

Possibility to implement threshold functions is due to negative differential resistance (NDR) that is a natural property of some circuits and devices. Unlike resistors or transistors, that have non decreasing current to voltage (I-V) characteristic, circuits that feature negative differential resistance have a part of their

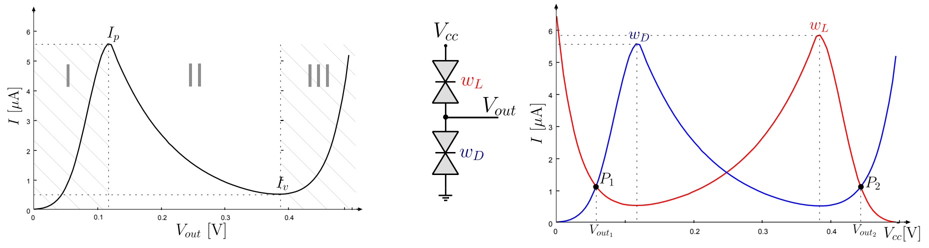


Fig. 1. Current to voltage (I-V) characteristic of single NDR device (left) and I-V characteristic of a simple circuit of two NDRs connected serially (right)

I-V characteristic decreasing. It means that in some part of I-V characteristic current that flows through the device decreases as the bias voltage (V_{cc}) increases. As a result a stable operating point of NDR-based circuit can be located in one of two positive differential resistance regions enabling circuit to switch between them. Moreover, this can by further extended to circuits with three or more stable operating points that are capable of implementing multivalued logic functions [12].

First circuits with NDR property were proposed in 70's and were constructed using CMOS technology [13,14]. They improved as a CMOS technology evolved but unfortunately several obstacles (e.g. large power consumption) were never eliminated and cause such structures to be out of practical interest. Situation has changed at the end of XX-th century when technology limits of traditional CMOS had become a serious threat to further development. This made scientists to look for a new type of electronic devices that can be fabricated in nanotechnology. They have come with different types of new transistors, tunnelling diodes and other devices and realised that some of them have a natural property of negative difference resistance [1,15]. Resonant tunnelling diodes (RTDs) are one of such device that have received a lot of focus resulting in a number of papers presenting their ability to implement different boolean functions [10,16] and constructing complex functional units [11].

Although many researchers have studied capabilities of NDR devices they have mostly focused on technological aspects of device fabrication and NDR-based circuit development. Less attention was paid to future application and circuit synthesis, especially that some problems have not been addressed so far. There are several NDR gate structures that differ in parameters and properties, however, all of them have one thing in common – branches of the circuit are always switched on with a non complemented signals. This is due to the technology limits which disallow to use a complementary transistor pair. In other words, only transistors that are switched on with a logic **1** state can be fabricated. Two gate structures have received a lot of attention in recent years – threshold gates (TG) and generalised threshold gates (GTG) [16,17]. The difference between TG and GTG is that GTG uses positive unate functions $N_i(\mathbb{X}^n)$ (i.e. negation free

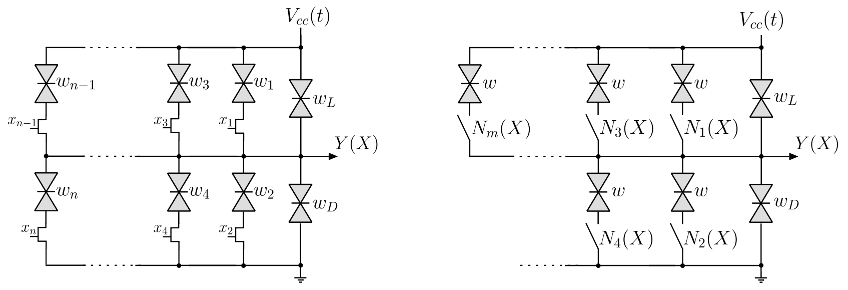


Fig. 2. General structure of TG (left) and GTG (right) circuit. Note that the difference is in complexity of functions switching circuit branches and their weights. In both circuits only non complementary input signals are used.

functions) to switch on/off branches of the circuit instead of using input signals directly. Precisely TG implements simple threshold function

$$Y(\mathbb{X}^n) = \begin{cases} 0 & \text{iff } \sum_{i=1}^{n} w_i x_i < T, \\ 1 & \text{otherwise }, \end{cases} \tag{1}$$

while GTG implements

$$Y(\mathbb{X}^n) = \begin{cases} 0 & \text{iff } \sum_{i=1}^{m} w_i N_i(\mathbb{X}^n) < T, \\ 1 & \text{otherwise }, \end{cases} \tag{2}$$

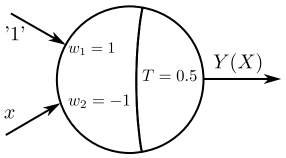
where $N_i(\mathbb{X}^n)$ is a positive unate functions of some input variables and $T = w_D - w_L$. Weights w_i in both TG and GTG circuits are determined by parameters of NDR devices and are positive for upper and negative for lower branches. Since TG element implements linearly separable function thus TG circuit can only implement some functions (e.g. AND, OR, NOT) while implementing more complex function is impossible. Capabilities of GTG circuit are enhanced as such circuits can implement more complex functions. Artificial neurons of GTG type were already proposed in [9], however this paper only briefly analyse such ANN while there was no detailed analysis on its capabilities. As we will present in next section a single GTG circuit, and consequently a GTG based ANN, can realise any Boolean function.

3 Implementing Boolean Functions with Threshold Elements

A well-known result from switching theory by Kohavi [5] is that any Boolean function of n variables can be realised using a two layer switching network assuming that input variables and their complements are available. It follows that there exist a two layer ANN consisting of neurons, implementing linear threshold function over the wighted sum of inputs (TG elements), that implements any Boolean function of n variables. Moreover, there is at most $2^n - 1$ neurons, implementing AND function, in the first (hidden) layer and a single neuron, implementing OR function, in the second (output layer). If we use TG elements to implement ANN where complementary signals are not available (which is the case for NDR devices) then ANN has to consist of three layers in order to implement all Boolean functions. This is formally stated in the following lemma.

Lemma 1. *Three layer artificial neural network implemented with NDR-based TG elements implements any Boolean function $Y(\mathbb{X}^n)$ of n variables.*

Proof. If we use a TG element with two inputs x_1, x_2 where x_1 is constantly equal 1 and x_2 is an input variable then setting weights to be equal 1 and -1 respectively, and threshold $T = 0.5$ yields the TG element to implement NOT function (fig. 3). It follows that a first layer of a three layer ANN, consisting of



x	$Y(X)$
0	1
1	0

Fig. 3. Symbol of a TG element implementing NOT function

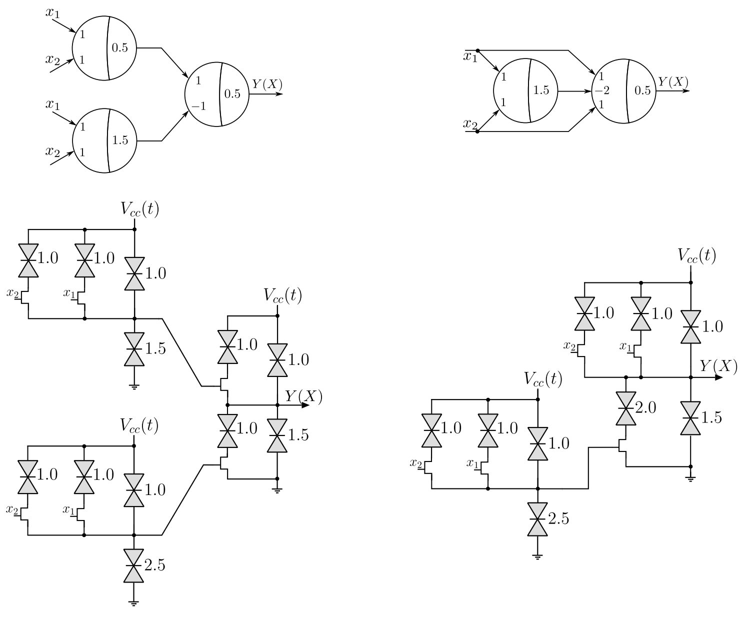


Fig. 4. Example of two artificial neural networks implementing 2 variable XOR function using 3 (left) and 2 (right) NDR-based TG elements

n nodes, computes complements of input variables that are used in successive layers to implement the Boolean function as stated by Kohavi [5].

While using three layer circuit of NDR-based NDR elements enables to implement any Boolean function in many cases simpler ANN structures can be used. For example, simple XOR function

$$Y(\mathbb{X}^2) = x_1 \oplus x_2 = x_1\overline{x_2} + \overline{x_1}x_2 \tag{3}$$

can be implemented using two layer network with three or even two neurons [18]. It follows that using TG elements one can construct an ANN to implement a given Boolean function. Figure 4 presents two implementations of two variable XOR function (3) implemented with three and two threshold elements and corresponding ANN structures.

3.1 Capabilities of Generalised Threshold Gates

There are only a few papers that comment on GTG circuits implementing different boolean logic functions [3,11,16] but they focus on feasibility and physical parameters rather then general capabilities of such structures.

A single GTG gate implements threshold function over the weighted sum of unate functions (2). Since threshold and weights depend on parameters of NDR elements thus we may assume that all NDR elements in GTG gate (except for w_L and w_D) have the same parameters. This allows to set threshold to 0.5 and make all weights equal to one and minus one depending on whether they correspond to upper or lower branches. Moreover we may select functions $N_i(\mathbb{X}^n)$ in such a way that they are positively unate and every function with index i is implied by any successive function with index $j > i$, that is

$$N_j(\mathbb{X}^n) \Rightarrow N_i(\mathbb{X}^n) \text{ for any } j > i. \tag{4}$$

The following properties of $N_i(\mathbb{X}^n), N_j(\mathbb{X}^n)$ functions follow from eq. (4):

$$N_j(\mathbb{X}^n)N_i(\mathbb{X}^n) = N_j(\mathbb{X}^n), \tag{5}$$
$$N_j(\mathbb{X}^n) + N_i(\mathbb{X}^n) = N_i(\mathbb{X}^n), \tag{6}$$
$$\overline{N_j(\mathbb{X}^n)}N_i = N_j(\mathbb{X}^n) \oplus N_i(\mathbb{X}^n). \tag{7}$$

If we use successive functions $N_i(\mathbb{X}^n)$ to alternately control upper and lower branches then the resulting function implemented by GTG gate (2) simplifies to

$$Y(\mathbb{X}^n) = \begin{cases} 0 & \text{iff } \sum_{i=1}^{m}(-1)^{i-1}N_i(\mathbb{X}^n) < 0.5, \\ 1 & \text{otherwise .} \end{cases} \tag{8}$$

Observe that term

$$\sum_{i=1}^{m}(-1)^{i-1}N_i(\mathbb{X}^n) \tag{9}$$

is a real valued sum that is only equal to 0 or 1 (since $N_i(\mathbb{X}^n)$ is a Boolean function) so this sum represents a Boolean function. Since we have selected $N_i(\mathbb{X}^n)$ in such a way that $N_j(\mathbb{X}^n) \Rightarrow N_i(\mathbb{X}^n)$ for any $j > i$ thus the following equalities hold:

$$\begin{aligned} N_1(\mathbb{X}^n) - N_2(\mathbb{X}^n) &= N_1(\mathbb{X}^n) \oplus N_2(\mathbb{X}^n) \\ N_1(\mathbb{X}^n) - N_2(\mathbb{X}^n) + N_3(\mathbb{X}^n) &= N_1(\mathbb{X}^n) \oplus N_2(\mathbb{X}^n) \oplus N_3(\mathbb{X}^n) \\ &\vdots \\ \sum_{i=1}^{m}(-1)^{i-1} N_i(\mathbb{X}^n) &= \bigoplus_{i=1}^{n} N_i(\mathbb{X}^n) \end{aligned}$$

It follows that the GTG circuit implements function

$$Y(\mathbb{X}^n) = \begin{cases} 0 & \text{iff } \bigoplus_{i=1}^{m} N_i(\mathbb{X}^n) = 0, \\ 1 & \text{otherwise }, \end{cases} \tag{10}$$

or equivalently

$$Y(\mathbb{X}^n) = \bigoplus_{i=1}^{m} N_i(\mathbb{X}^n). \tag{11}$$

Equation (11) states that a GTG gate implements the XOR sum of positively unate functions $N_i(\mathbb{X}^n)$ such that $N_j(X) \Rightarrow N_i(\mathbb{X}^n)$ for any $j > i$. However, it is not known what is the set of Boolean functions that can be implemented in such a way and how many terms the XOR sum consist of. This is stated in the following theorem.

Theorem 1. *Any Boolean function $Y(\mathbb{X}^n)$ of n variables can be implemented as an XOR sum of unate functions $N_i(\mathbb{X}^n)$ such that $N_j(X) \Rightarrow N_i(\mathbb{X}^n)$ for any $j > i$. The XOR sum consist of at most n elements.*

Proof. First, observe that there are no complementary variables in sum-of-product (SOP) representation of positively unate functions $N_i(\mathbb{X}^n)$. It is so, since positively unate function is non decreasing one so when any input signal change from logic **0** to **1** then $N_i(\mathbb{X}^n)$ either stays unchanged or switches to **1**.

Second, note that for any Boolean function $Y_i(\mathbb{X}^n)$ there exist a function $N_i(\mathbb{X}^n)$ that is positively unate and $Y_i(\mathbb{X}^n) \Rightarrow N_i(\mathbb{X}^n)$. It is so, since $N_i(\mathbb{X}^n) = \mathbf{1}$ is a positively unate function and any $Y_i(\mathbb{X}^n) \Rightarrow \mathbf{1}$. For most functions $Y_i(\mathbb{X}^n)$ there are several functions $N_i(\mathbb{X}^n)$ and the smallest one (in terms of number of input vectors $\mathbb{X}^n$ for which $N_i(\mathbb{X}^n) = \mathbf{1}$) can be found. This can be easily done through removing all complementary variables form the SOP representation of $Y_i(\mathbb{X}^n)$ (note that $Y_i(\mathbb{X}^n)$ is any Boolean function, so it might not be positively unate).

From the above analysis we may conclude that for any Boolean function $Y_0(\mathbb{X}^n)$ we may found the smallest unate function $N_1(\mathbb{X}^n)$ and represent $Y_0(\mathbb{X}^n)$ as

$$Y_0(\mathbb{X}^n) = N_1(\mathbb{X}^n) \oplus Y_1(\mathbb{X}^n). \tag{12}$$

If $Y_1(\mathbb{X}^n)$ is not positively unate than we may found the smallest, positively unate function $N_2(\mathbb{X}^n)$ such that $Y_1(\mathbb{X}^n) \Rightarrow N_2(\mathbb{X}^n)$ and

$$Y_1(\mathbb{X}^n) = N_2(\mathbb{X}^n) \oplus Y_2(\mathbb{X}^n). \tag{13}$$

Since $Y_0(\mathbb{X}^n)Y_1(\mathbb{X}^n) = \mathbf{0}$ and $N_2(\mathbb{X}^n)$ is the smallest unate function for $Y_1(\mathbb{X}^n)$ thus $N_2(\mathbb{X}^n) \subset N_1(\mathbb{X}^n)$ or equivalently $N_2(\mathbb{X}^n) \Rightarrow N_1(\mathbb{X}^n)$ and $N_2(X)N_1(X) \neq N_1(X)$. Repeating the above procedure subsequently to successive functions $Y_i(X)$ we are able to represent $Y_0(\mathbb{X}^n)$ as

$$Y_0(\mathbb{X}^n) = \bigoplus_{i=1}^{k} N_i(\mathbb{X}^n) \oplus Y_k(\mathbb{X}^n). \tag{14}$$

Observe that it follows from $Y_{i-1}(\mathbb{X}^n) \Rightarrow N_i(\mathbb{X}^n)$ that

$$Y_{i-1}(\mathbb{X}^n) \subseteq N_i(\mathbb{X}^n) \tag{15}$$

$$Y_i(\mathbb{X}^n) \subset N_i(\mathbb{X}^n) \tag{16}$$

$$Y_{i-1}(\mathbb{X}^n)Y_i(\mathbb{X}^n) = \mathbf{0} \tag{17}$$

Since $N_j(\mathbb{X}^n) \subset N_i(\mathbb{X}^n)$ for any $j > i$ and there are no complementary signals in SOP representation of these functions thus products in $N_j(\mathbb{X}^n)$ are over the larger number of variables then in $N_i(\mathbb{X}^n)$. Therefore, if we assume $N_1(\mathbb{X}^n)$ is a sum of single variables then products in $N_2(\mathbb{X}^n)$ are over two or more input variables. Consequently, $N_k(\mathbb{X}^n)$, for some $k \leq n$, is a single product of all variables (i.e. $N_k(\mathbb{X}^n) = x_1x_2 \ldots x_n$). Since $Y_i(\mathbb{X}^n) \subset N_i(\mathbb{X}^n)$ (16) thus $Y_k(\mathbb{X}^n) \subset x_1x_2 \ldots x_n$ and so it has to be equal $\mathbf{0}$.

Therefore, it follows from (14) that any n variable Boolean function $Y_0(\mathbb{X}^n)$ can be represented as an XOR sum of at most n positively unate functions $N_i(\mathbb{X}^n)$. That is

$$Y_0(\mathbb{X}^n) = \bigoplus_{i=1}^{k} N_i(\mathbb{X}^n), \tag{18}$$

where $k \leq n$.

The above theorem states that GTG gate can implement any Boolean function. Consequently, ANN based on GTG gates would be possible to implement any Boolean function using a single GTG-neuron.

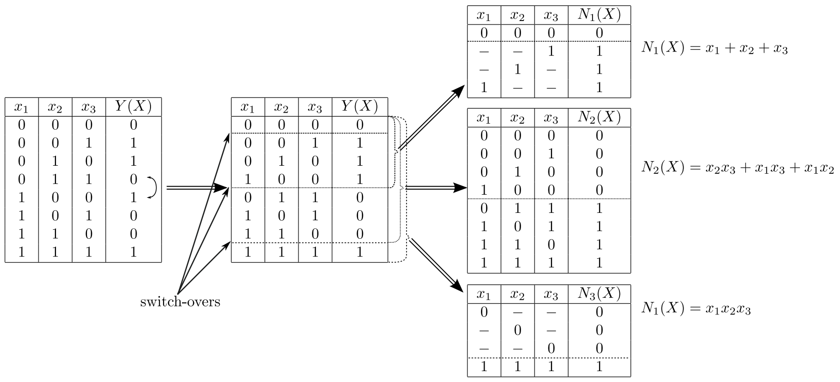
3.2 GTG Gate Synthesis

Synthesis algorithm can be derived directly from the proof of theorem 1. However, implementing it that way requires formal definition of minimal positively unate functions, and a method how to find such for a given boolean function. In general the algorithm recursively searches for the smallest, positively unate functions $N_i(\mathbb{X}^n)$ for given $Y_{i-1}(\mathbb{X}^n)$ and computes $Y_i(\mathbb{X}^n) = Y_{i-1}(\mathbb{X}^n) \oplus N_i(\mathbb{X}^n)$. This is repeated until $Y_k(\mathbb{X}^n)$ calculated equals $\mathbf{0}$. The algorithm outputs all unate $N_i(\mathbb{X}^n)$ for $i = 1, 2, \ldots, k$ [10].

Algorithm 1. Synthesis of GTG gates for ANN

```
Require: $n$–variable Boolean function $Y(\mathbb{X}^n)$
Ensure: $w_L, w_D$ determining threshold $T$ and $N_i(\mathbb{X}^n)$ functions
 1: if $Y(0^n) = 0$ then $w_L = 1.0,\ w_D = 1.5 \ \Rightarrow\ T = w_D - w_L = 0.5$
 2: else $w_L = 1.5,\ w_D = 1.0 \ \Rightarrow\ T = w_D - w_L = -0.5$,
 3: construct the truth table for $Y(\mathbb{X}^n)$ – $\mathcal{Y}$,
 4: sort $\mathcal{Y}$ by the hamming weight of $\mathbb{X}^n$ vectors and function value so that the number
    of switches $k$ from 0 to 1 and vice verse is minimised,
 5: set $i = 1$,
 6: repeat
 7:    construct the truth table for function $N_i(\mathbb{X}^n)$ – $\mathcal{N}_i$:
         • take rows from $\mathcal{Y}$ starting from first row up to the row that precedes $i$-th
           change of the $Y(\mathbb{X}^n)$ function value; set $N_i(\mathbb{X}^n)$ =0 for these rows,
         • complete the table using remaining input vectors; set $N_i(\mathbb{X}^n)$ =1,
 8:    minimise and store function $N_i(\mathbb{X}^n)$,
 9:    set $i = i + 1$,
10: until $i \leq k$
```

Fortunately, there is a simpler version of the algorithm that is similar to threshold network synthesis algorithms and is efficient for small n (e.g. $n < 10$). Algorithm 1 draws on the truth table of the given function $Y(\mathbb{X}^n)$ that is sorted by Hamming weights of input vectors first and function values afterwards. This ensures that the resulting truth table will minimise the number of switch-overs of function value – this is known from other ANN synthesis algorithms. Later on i-th switch-over is used to construct a new truth table for $N_i(\mathbb{X}^n)$ function. This is constructed in such a way that $N_i(\mathbb{X}^n)$ =**0** for all input vectors that fall before the i-th switch-over, and $N_i(\mathbb{X}^n)$ =**1**, otherwise. Finally, resulting functions $N_i(\mathbb{X}^n)$ are minimised. Example of such synthesis is presented on fig. 5 where XOR function of three variables is synthesised.



x_1	x_2	x_3	$Y(X)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

x_1	x_2	x_3	$Y(X)$
0	0	0	0
0	0	1	1
0	1	0	1
1	0	0	1
0	1	1	0
1	0	1	0
1	1	0	0
1	1	1	1

x_1	x_2	x_3	$N_1(X)$
0	0	0	0
–	–	1	1
–	1	–	1
1	–	–	1

$N_1(X) = x_1 + x_2 + x_3$

x_1	x_2	x_3	$N_2(X)$
0	0	0	0
0	0	1	0
0	1	0	0
1	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1

$N_2(X) = x_2x_3 + x_1x_3 + x_1x_2$

x_1	x_2	x_3	$N_3(X)$
0	–	–	0
–	0	–	0
–	–	0	0
1	1	1	1

$N_1(X) = x_1x_2x_3$

Fig. 5. Successive steps of XOR$_3$ synthesis for GTG circuit

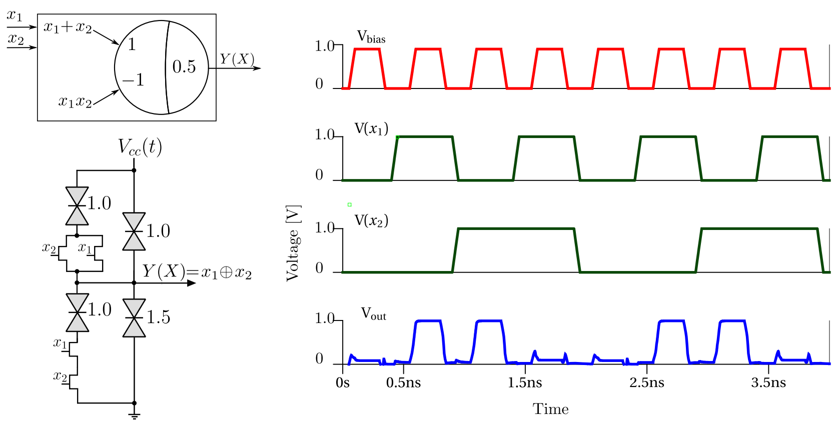


Fig. 6. Model and structure of GTG gate implementing XOR$_2$ (left) and a transient analysis of this circuit (right)

We have implemented the algorithm and have run it for different Boolean functions. It was verified for Boolean functions of up to $n < 10$ variables since its complexity grows exponentially with n. However, it is very unlikely that Boolean functions of more then 10 or 15 inputs will be implemented in single gate structure. This is due to the technological limits and parasitic effects that nanocircuit will have to cope with. Therefore, more complex functions will probably be first decomposed and then implemented as a network of NDR-based gates of small number of inputs.

Functions $N_i(\mathbb{X}^n)$ outputted from the synthesis algorithm were later used to model the circuit in SPICE software and simulated to verify whether GTG gate implements the given function. Figure 6 presents a logic model of GTG gate implementing XOR$_2$ function, gate structure and transient analysis from SPICE.

4 Conclusions

It seems like after almost 50 years there is a real chance to take advantages of threshold gates and logic for constructing Boolean logic circuits and electronic devices. Despite the fact that, nanotechnology is still immature there is a great attention paid to this kind of devices as they are thought to be future of electronic circuits. Interesting properties, feasibility at nanometer integration scale, high operational frequency and low power consumption are features that ensure these kind of devices will gain a lot of attention in next years. Nevertheless, some technological problems that threshold gates still experience nowadays, engineers were never closer of making this kind of hardware an of the shelf technology. Irrespective of technology details, that will be used to construct NDR devices in future, the theory of GTG circuit synthesis will work and will enable to construct complex Boolean functions.

This paper presents state of the art in contemporary efforts for efficient hardware implementation of threshold function. It is very likely that technical obstacles will be solved in comming years enabling to use this type of nanocircuits to construct application specific neural networks. We have briefly presented capabilities of NDR-based circuits and gave a simple synthesis algorithm. It requires memory that grows exponentially with number of variables, however, it is independent of the function representation.

Acknowledgments

This work was supported by grant no. N N516 451538 from the Polish Ministry of Science and Higher Education.

References

1. Bergman, J., Chang, J., Joo, Y., Matinpour, B., Laskar, J., Jokerst, N., Brooke, M., Brar, B., Beam, E.: RTD/CMOS nanoelectronic circuits: thin-film InP-based resonant tunneling diodes integrated with CMOS circuits. IEEE Electron Device Letters 20(3), 119–122 (1999)
2. Kelly, P., Thompson, C., McGinnity, T., Maguire, L.: Investigation of a programmable threshold logic gate array. In: 9th International Conference on Electronics, Circuits and Systems, vol. 2, pp. 673–676 (2002)
3. Pettenghi, H., Avedillo, M.J., Quintana, J.M.: Using multi-threshold threshold gates in rtd-based logic design: A case study. Microelectron. J. 39(2), 241–247 (2008)
4. Muroga, S.: Threshold logic and its applications. John Wiley & Sons, Chichester (1971)
5. Kohavi, Z.: Switching and Finite Automata Theory. McGraw-Hill College (1978)
6. Fausett, L.V.: Fundamentals of artificial neural networks. Prentice-Hall, Englewood Cliffs (1993)
7. Schmitt, M.: On computing boolean functions by a spiking neuron. Annals of Mathematics and Artificial Intelligence 24(1-4), 181–191 (1998)
8. Anthony, M.: Boolean functions and artificial neural networks. Technical report, Report Department of Mathematisc and Centre for Discrete and Applicable Mathematics, The London School of Economics and Political Science (2003)
9. Subirats, J., Gómez, I., Jerez, J., Franco, L.: Optimal synthesis of boolean functions by threshold functions. In: Kollias, S.D., Stafylopatis, A., Duch, W., Oja, E. (eds.) ICANN 2006. LNCS, vol. 4131, pp. 983–992. Springer, Heidelberg (2006)
10. Bawiec, M., Nikodem, M.: Generalised threshold gate synthesis based on and/or/not representation of boolean function. In: Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific, pp. 861–866 (January 2010)
11. Pettenghi, H., Avedillo, M., Quintana, J.: Improved nanopipelined rtd adders using generalized threshold gates. IEEE Transactions on Nanotechnology (2009)
12. Berezowski, K.S., Vrudhula, S.B.K.: Multiple-valued logic circuits design using negative differential resistance devices. In: ISMVL 2007: Proceedings of the 37th International Symposium on Multiple-Valued Logic, Washington, DC, USA, p. 24. IEEE Computer Society, Los Alamitos (2007)

13. Wu, C., Lai, K.N.: Integrated λ-type differential negative resistance mosfet device. IEEE Journal of Solid-State Circuits 14(6), 1094–1101 (1979)
14. Gan, K.J., Hsiao, C.C., Wang, S.Y., Chiang, F.C., Tsai, C.S., Chen, Y.H., Kuo, S.H., Chen, C.P., Liang, D.S.: Logic circuit design based on mos-ndr devices and circuits fabricated by cmos process. In: Proceedings of Fifth International Workshop on System-on-Chip for Real-Time Applications, pp. 392–395 (20-25, 2005)
15. ITRS: Emerging research devices. Technical report, International Technology Roadmap for Semiconductors (2009)
16. Berezowski, K.S.: Compact binary logic circuits design using negative differential resistance devices. Electronics Letters 42(16), 5–6 (2006)
17. Bawiec, M.A.: Resonant tunnelling diode-based circuits: Simulation and synthesis. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) Computer Aided Systems Theory - EUROCAST 2009. LNCS, vol. 5717, pp. 873–880. Springer, Heidelberg (2009)
18. El-Bakry, H.M., Atwan, A.: Simplification and Implementation of Boolean Functions. International Journal of Universal Computer Science 1(1), 41–50 (2005)

Solving No-Wait Flow Shop Scheduling Problems by a Hybrid Quantum-Inspired Evolutionary Algorithm

Tianmin Zheng[1] and Mitsuo Yamashiro[2]

[1] SoftAgency Co., Ltd. Oyama, Tochigi, 323-0824, Japan
tenminsa@gmail.com
[2] Ashikaga Institute of Technology, Ashikaga, Tochigi, 326-8558, Japan
yamashiro@ashitech.ac.jp

Abstract. This paper is the first to consider the hybrid quantum-inspired evolutionary algorithm (HQEA) on the no-wait permutation flow shop scheduling problem (PFSP) for minimizing the makespan. In this HQEA, the quantum chromosomes are encoded by using the quantum rotating angle and a simple but efficient converting mechanism for determining job sequence is proposed for the representation of solution firstly. Then, we adopt differential operation to perform the updating of quantum gate and the local search to perform exploitation in the promising permutation-based solutions. We make the simulations on famous benchmarks and the comparisons with other state-of-the-art approaches demonstrate the effectiveness of the proposed HQEA for no-wait flow shop scheduling problem.

Keywords: quantum-inspired evolutionary algorithm; no-wait; permutation flow shop; differential operation; local search.

1 Introduction

As an important part of the combinatorial optimization problems, the flowshop scheduling problem (FSP) is a class of scheduling problem which has been widely studied by many workers due to its importance both in academic and engineering fields. A FSP containing the same processing sequence of jobs for all machines is called as permutation FSP (PFSP). In this paper, we consider the no-wait PFSP with the criterion of minimizing the makespan which is widely adopted in the chemical processing, food processing and concrete ware production systems. In the no-wait PFSP, the word no-wait means the processing of each job has to be continuous and a job must be processed from the time of start to completion without any interruption. Therefore, the start of a job on the given machine must be delayed in order to meet the no-wait requirement which is different from the common PFSP.

Given the processing time $p(j, k)$ for job j on machine k in a $n\times m$ no-wait PFSP, the completion time of job j on machine k must be equal to the earliest start time on machine k+1. Suppose $\pi = \{J_1, J_2,\ldots,J_n\}$ be any a processing sequence of all jobs for no-wait PFSP, and let $d(J_{i-1}, J_i)$ be the minimum delay on the first machine between the start of job J_i and $J_{i\text{-}1}$ when job J_i is directly processed after job $J_{i\text{-}1}$. The minimum delay can be computed from the following expression:

$$d(J_{i-1},J_i)=p(J_{i-1},1)+\max\left[0,\max_{2\le k\le m}\left\{\sum_{h=2}^{k}p(J_{i-1},h)-\sum_{h=1}^{k-1}p(J_i,h)\right\}\right] \tag{1}$$

Then the maximum completion time (makespan) can be calculated as:

$$C_{\max}(\pi)=\sum_{i=2}^{n}d(J_{i-1},J_i)+\sum_{k=1}^{m}p(J_n,k) \tag{2}$$

The no-wait PFSP with the objective of minimizing the makespan is to find the permutation π^* in the set of all permutations Π satisfies the following criterion:

$$C_{\max}(\pi^*)\le C_{\max}(\pi)\qquad \forall\pi\in\Pi \tag{3}$$

The no-wait flowshop scheduling problem with more than two machines is strong NP-hard [1]. By now, the heuristic and meta-heuristic algorithms achieve global or sub-optimal optima within acceptable time range is most popular for dealing with the no-wait PFSP. These approaches are initiated from a set of solutions and try to improve these solutions by using some strategies or rules which include constructive heuristic [2, 3], simulated annealing (SA) [4], genetic algorithm (GA) [4], hybrid GA and SA (GASA) [5], variable neighborhood search (VNS) [5], tabu search (TS) [6], hybrid and discrete particle swarm optimization (PSO) [7, 8]. Recently, Han and Kim [9] proposed the quantum-inspired evolutionary algorithm (QEA) for knapsack problem. However, due to its encoding and decoding scheme, the QEA can't directly be applied to PFSP and the research of production scheduling based on QEA is just at beginning. Wang [10] is the first to adopt QEA to minimize makespan of PFSP. Quite recently, Gu [11] proposed a quantum genetic scheduling algorithm for stochastic FSP with the random breakdown, Niu [12] put forward a quantum-inspired immune algorithm for hybrid FSP with makespan criterion and Zheng [13] adopted a quantum differential evolutionary algorithm to solve the permutation flow shop scheduling.

In this paper, we propose a novel hybrid QEA for no-wait PFSP, especially to develop a hybrid strategy by combining Q-bit based search, differential operation and local search to achieve better performance. The organization of the remaining content is as follows. We will introduce each important part of the proposed quantum-inspired evolutionary algorithm as well as its implementation in section 2. In section 3, we make the simulation and comparisons of proposed HQEA with other algorithms for no-wait PFSP. Finally, we will make a brief conclusion to end this paper in section 4.

2 Hybrid QEA for No-Wait PFSP

The basic quantum-inspired evolutionary algorithm (QEA) is proposed by Han and Kim [9] firstly, and they adopted this new algorithm to solve the knapsack problem. The mechanism of QEA is based on the concepts and principles of quantum computing, such as the quantum bit and the superposition of states. The QEA can explore the search space with a smaller number of individuals and exploit the search space for a global solution within a short span of time. However, QEA is not a quantum algorithm, but a novel evolutionary algorithm (EA). Like any other EAs, QEA is also characterized by the representation of the individual, the evaluation function, and the population diversity. Inspired by the concept of quantum computing, QEA is designed with a novel Q-bit representation, a Q-gate as a variation operator, and an observation process based on the Q-bits.

2.1 The Encoding and Decoding Strategy for No-Wait PFSP

According to the principles of the basic QEA, each Q-bit has two probability amplitudes, so the quantum chromosome with the problem dimension of *m* is composed by two strings of probability amplitudes shown in equation (4).

$$q=\begin{bmatrix}\alpha_1 & \alpha_2 & \cdots & \alpha_m\\ \beta_1 & \beta_2 & \cdots & \beta_m\end{bmatrix} \tag{4}$$

In this paper, we suppose the quantum chromosome is represented as:

$$q=[\theta_1 \quad \theta_2 \quad \cdots \quad \theta_m] \tag{5}$$

where θ_i is the quantum rotating angle with the range of [0, π/2]. We operate on the rotating angle and update it with the updating operation.

Since the updating operation is the key of the QEA and the essence of the updating procedure is to influence on the probability amplitude by changing the value of rotating angle, so we can directly practice on the quantum chromosomes represented in rotating angle. This provides a more effective way to deal with the Q-bits. Also, the encoding of quantum chromosome by using rotating angle can simplify the operations performed on them such as the differential operations by using only one variable. Therefore, we consider using the rotation angle to encode the quantum chromosome is more suitable for PFSP than probability amplitude which has been adopted to solve the PFSP by Wang [10], the stochastic FSP by Gu [11] and hybrid FSP by Niu [12].

As for the decoding process of the quantum chromosome presented in the form of rotating angle, since the solution to the PFSP is the technological order of all the jobs, therefore, we should convert the quantum chromosome which is represented by Q-bit into job sequences. In the decoding scheme adopted by Wang [10] and Gu [11], the representation needs several conversions (Q-bit chromosome → binary chromosome → decimal chromosome → job order) and the computation is complicated when the problem scale becomes larger. We put forward a much simpler mechanism for conversion which is an innovation in this paper. We define the rotating angles of 1,2,…,*n* are $[\theta_1, \theta_2, \ldots, \theta_n]$, that the probability amplitude of job *i* is $[\cos\theta_i, \sin\theta_i]$, then determine the job sequence according to the following procedure:

Step 1: Obtain one quantum chromosome $q_i = [\theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,n}]$ from the Q-bit based population, calculate $temp_i = [\cos\theta_{i,1}, \cos\theta_{i,2},\ldots,\cos\theta_{i,n}]$ and initiate two empty arrays *first*() and *last*().

Step 2: Generate a random number η between [0, 1] and compare it with $\cos\theta_{i,job}$ where $job\in[1, n]$. If $\cos\theta_{i,job}>\eta$, put job into *first*(), else put job into *last*(). Repeat until all Q-bits in q_i are operated.

Step 3: Combine these two arrays *first*() and *last*() to one array *permutation*(), the element in *permutation*()is the processing job sequence for PFSP.

For example, we have q_i = [0.87,0.68,0.15,0.42,1.38,1.09], so $temp_i$ = [0.64,0.77, 0.98,0.91,0.18, 0.46]. Then generate η as 0.76 which is larger than $\cos\theta_{i,1}$, so we put '1' into *last*(). We continue generate η as 0.37 and put '2' into *first*() since it is smaller than $\cos\theta_{i,2}$. After we operated on all 6 Q-bits, we have *first*()= [2 3 6] and *last*()=[1 4 5]. By combining these two arrays, we have *permutation*() = [2 3 6 1 4 5] which is the processing order for these 6 jobs.

Here, suppose we have a scheduling problem with the scale of 100 jobs and we can make a comparison like this way: by the method proposed by Wang [10] and Gu [11], since $2^6<100<2^7$, so the length of the quantum chromosome should be 100×7=700 at least, and three conversions are needed to get the job sequences which requires lots of calculating time. While, by our method, we just need a quantum chromosome with length of 100 and practice the conversion only once. Thus, the representation of the solution we proposed simplifies the decoding procedure greatly and can provide a more effective way to deal with no-wait PFSP. On the other hand, in [13], the quantum chromosomes are also encoded in the form of rotating angle and a conversion mechanism called largest rotating angle value (LRAV) rule is adopted to make the decoding procedure. While, we notice that the LRAV rule is simply performed by sorting the rotating angle value in the quantum chromosomes, so it loses the diversity of the Q-bit and can not make full use of the probability amplitude information. By our decoding strategy, the η is generated randomly and the job sequence is determined by comparing the η and the probabilistic amplitude of each Q-bit which will provide us with good population diversity.

2.2 Quantum Updating by Differential Operation

Due to its simple concept and easy implementation of the differential evolution (DE) proposed by Storn and Price [14], we adopt the differential operation to update the quantum chromosomes. The differential operations include mutation, crossover and selection operation work on real number differential vectors and have excellent ability of overall search ability.

In our HQEA, Suppose the quantum population $Q_g = [q_{1,g}, q_{2,g},\ldots, q_{n,g}]$ (n is the population scale, g is the current evolutionary generation), individual $q_{i,g} = [\theta_{i,1,g}, \theta_{i,2,g},\ldots, \theta_{i,m,g}]$ (m is the dimension of the problem, $i\in[1, n]$). Suppose $v_{i,g+1}$ is the corresponding individual obtained by practicing the mutation operator on individual $q_{i,g}$, and the mutation operator we adopted in this paper works as:

$$v_{i,g+1} = q_{r1,g} + F(q_{r2,g} - q_{r3,g}) \tag{6}$$

where $r1, r2, r3 \in [1, n]$ and $r1 \neq r2 \neq r3 \neq i$; $q_{r1,g}$ is called father basic vector, ($q_{r2,g} - q_{r3,g}$) is called father differential vector; F is a real and constant factor which controls the amplification of the differential variation.

In order to increase the diversity of the parameter vectors, we also use the $u_{i,j,g+1}$ ($j \in [1, m]$) vector which is obtained by practicing kinds of crossover operation between

$q_{i,g}$ and mutative individual $v_{i,g+1}$ obtained by equation (6). The bin crossover we will use in this paper is shown in equation (7):

$$u_{i,j,g+1}\begin{cases} v_{i,j,g+1} & for \quad rand < CR \quad or \quad j = Jrnd \\ \theta_{i,j,g} & otherwise \end{cases} \tag{7}$$

where *CR* is the crossover factor and *Jrnd* is chosen randomly from the interval [1, *m*].

Since the quantum chromosomes are encoded in the form of quantum angle, so the differential operations are directly practiced on the quantum angle and can provide the updating with excellent overall search ability and diversity.

2.3 Local Search

By adopting the proposed decoding mechanism, the Q-bits based population can be converted to permutative-based solution for scheduling, so the local search can be easily embedded to develop effective hybrid algorithms. In this paper, a simple local search method based on *insert* neighborhood is adopted to further improve the solution quality. We obtain the global best chromosome *Best_g* and suppose *n* is the job number for a special problem. After initializing iteration counter $k = 0$, flag *found* = *false* and makespan of *Best_g* as *M*(*Best_g*), then the local search works as:

```
while (found==false) and ($k < n^{1/2}$) do
  generate i, remove i-th job in Best_g and obtain a partial sequence temp
  insert the removed job into the best position j(j≠i) in temp and calculate
      M(temp)
  if M(temp) < M(Best_g) then
   found=true
  else
   $k = k + 1$
  endif
end
```

As for the *insert* neighborhood based local search, Pan [8] proposed a speed-up method for calculating the makespan when we remove or insert a job in job sequence. This speed-up method will also be embedded into the local search in our paper.

2.4 The Main Procedure of HQEA

The main procedure of hybrid quantum-inspired evolutionary algorithm for no-wait permutation flow shop scheduling is described as follows:

Step 1: Initialize control parameters. Set the value of control parameters for differential operation and the maximum evolutionary generation t_{max}. Initialize iteration counter $t = 0$.

Step 2: Initialize the population. Determine the initial population $Pop_0 = [chrom_{0,1}, chrom_{0,2},..., chrom_{0,n}]$, where $chrom_{0,i} = [\theta_{0,1}, \theta_{0,2},..., \theta_{0,m}]$, *n* is the population scale, *m* is the number of jobs.

Step 3: Make the solution. Adopt the decoding strategy to make the solution for permutation-base problem from the Q-bits based population.
Step 4: Obtain objective values by evaluating Pop_0; store the best individual into $Best\_\theta$ and best job sequence into $Best\_g$.
Step 5: Perform the evolution.
Step 5.1: Updating. Update the Pop_{t-1} to Pop_t by using differential operations.
Step 5.2: Make the solution. Adopt the decoding strategy to make the solution for permutation-based problem from the Q-bits based population.
Step 5.3: Evaluation. Evaluate Pop_t and get makespan, compare with $Best_{t-1}$ and store the better one to $Best_t$. Update the $Best\_\theta$ and $Best\_g$.
Step 5.4: Local search. Practice the local search on $Best\_g$.
Step 6: Stopping condition check. If the stopping condition $t > t_{max}$ is met or the optimum is found, output the optimum; else $t \leftarrow t + 1$ and go to **step 5**.

3 Simulations and Comparisons

3.1 Experiment Settings

To test the performance of the proposed HQEA, computational simulation is carried out with 29 benchmarks from Carlier [15] and Reeves [16]. Thus far these problems have been used as benchmarks for study with different methods by many researchers. In order to make comparisons by using different methods, we adopt the relative percent error (RE) which is widely used in other literatures. The BRE, ARE and WRE stand for the best, average and worst relative percentage error to the optimal solution named C_{max^*}. After n replications of simulations, the BRE, ARE and WRE will be calculated as follows (the M is short for makespan):

$$BRE(ARE,WRE) = \sum_{r=1}^{n} \left(\frac{M_{best,avg,worst} - C_{max^*}}{C_{max^*}} \times 100 \right) \times \frac{1}{n} (\%) \tag{8}$$

Meanwhile, for the differential operations, we set the crossover factor $CR = 0.9$ and weight factor $F = 0.1$, and adopt the equation (6) and (7) to perform the evolution. For running of HQEA, we set population scale to be n (the number of jobs); set the maximum evolutionary generation t_{max} to be 500 and run the algorithm 20 times for each problem to make the discussion.

3.2 Comparisons of HQEA and HPSO

Firstly, to show the effectiveness of the proposed HQEA, we want to compare the HQEA with the HPSO developed by Liu [7] based on the Car and Rec benchmark problems. Here, the values of C_{max^*} for 8 Car problems are the optimal makespans or lower bound values known, while, for 21 Rec problems these values are provided by the famous RAJ algorithm [2]. The results are shown in Table 1.

From Table 1, we can see that the HQEA provides the better optimization performance than the HPSO. For the Car problems with small scale, HPSO found the optimal 5 out of 8 problems in every running, while HQEA can find 7. For the Rec problems with relatively large scale, the two hybrid algorithms show the significant improvement over RAJ algorithm which means the all the REs are much smaller than 0. Compared with the HPSO, the BRE, ARE and WRE our HQEA obtained are better than those obtained by HPSO for almost all of the instances. On the other hand, we notice the ARE and WRE values resulting from HQEA are also very close to the BRE. All of these demonstrate that the proposed HQEA is a novel, effective and robust algorithm for the no-wait permutation flowshop scheduling problems.

Table 1. Results of testing between HQEA and HPSO

P	N,M	C_{max*}	HPSO			HQEA		
			BRE	ARE	WRE	BRE	ARE	WRE
Car1	11,5	8,142	0.00	0.00	0.00	0.00	0.00	0.00
Car2	13,4	8,242	0.00	0.18	0.61	0.00	0.00	0.00
Car3	12,5	8,866	0.00	0.06	0.24	0.00	0.00	0.00
Car4	14,4	9,195	0.00	1.85	4.29	0.00	0.01	0.07
Car5	10,6	9,159	0.00	0.00	0.00	0.00	0.00	0.00
Car6	8,9	9,690	0.00	0.00	0.00	0.00	0.00	0.00
Car7	7,7	7,705	0.00	0.00	0.00	0.00	0.00	0.00
Car8	8,8	9,372	0.00	0.00	0.00	0.00	0.00	0.00
Rec01	20,5	1,590	−3.77	−3.39	−2.96	−4.03	−4.03	−4.03
Rec03	20,5	1,457	−6.59	−6.15	−3.36	−6.59	−6.59	−6.59
Rec05	20,5	1,637	−7.39	−7.15	−6.66	−7.70	−7.58	−7.45
Rec07	20,10	2,119	−3.63	−3.11	−2.31	−3.63	−3.62	−3.53
Rec09	20,10	2,141	−4.58	−4.26	−3.60	−4.62	−4.57	−4.54
Rec11	20,10	1,946	−3.34	−2.30	−1.28	−3.34	−3.33	−3.31
Rec13	20,15	2,709	−6.05	−5.47	−4.80	−6.05	−5.92	−5.83
Rec15	20,15	2,691	−6.02	−5.69	−4.91	−6.02	−6.02	−6.02
Rec17	20,15	2,740	−5.58	−5.42	−5.07	−5.58	−5.58	−5.58
Rec19	30,10	3,157	−9.15	−8.50	−6.46	−9.72	−9.34	−9.07
Rec21	30,10	3,015	−5.70	−5.33	−4.74	−6.43	−6.12	−5.85
Rec23	30,10	3,030	−10.80	−9.72	−8.65	−10.89	−10.19	−9.87
Rec25	30,15	3,835	−5.71	−5.17	−4.25	−6.31	−6.03	−5.72
Rec27	30,15	3,655	−6.13	−5.04	−4.13	−6.13	−5.56	−5.33
Rec29	30,15	3,583	−7.81	−6.93	−5.69	−8.15	−7.73	−7.54
Rec31	50,10	4,631	−5.92	−5.20	−4.51	−6.89	−7.32	−6.01
Rec33	50,10	4,770	−5.51	−4.08	−3.17	−7.01	−6.54	−6.21
Rec35	50,10	4,718	−6.02	−5.13	−3.98	−6.72	−6.54	−6.31
Rec37	75,20	8,979	−8.89	−8.20	−7.40	−10.39	−9.97	−9.65
Rec39	75,20	9,158	−6.79	−5.67	−4.26	−7.56	−7.21	−6,89
Rec41	75,20	9,344	−7.94	−6.77	−5.91	−9.28	−8.87	−8.65
AVE			−4.60	−4.02	−3.21	−4.93	−4.78	−4.54

3.3 Comparisons of HQEA with Other Existing Approaches

At last, we also make the comparison of HQEA with other existing approaches include the VNS and GASA by Aldowaian [5], the DS, DS+M, TS, TS+M and TS+MP by Grabowski [6] and DPSO by Pan [8]. Since the QDEA by Zheng [13] is similar to the proposed algorithm and has been adopted to solve the permutation FSP, we

Table 2. Comparisons of HQEA and other approaches

P	VNS	GASA	DS	DS+M	TS	TS+M	TS+MP	QDEA	DPSO	HQEA
Car1	0.70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Car2	0.20	0.00	0.62	0.62	0.00	0.00	0.00	0.00	0.00	0.00
Car3	0.00	0.00	0.08	0.08	0.00	0.00	0.00	0.00	0.00	0.00
Car4	1.60	0.00	2.77	2.77	0.00	0.00	0.00	0.00	0.00	0.00
Car5	3.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Car6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Car7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Car8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Rec01	−2.77	−3.96	−3.71	−3.58	−4.03	−3.96	−3.96	−4.03	−4.03	−4.03
Rec03	−4.32	−4.46	−3.43	−4.43	−6.59	−6.59	−6.59	−6.59	−6.59	−6.59
Rec05	−7.03	−6.90	−5.62	−5.62	−7.39	−7.64	−7.70	−7.70	−7.70	−7.70
Rec07	−2.31	−3.45	−1.09	−1.09	−3.63	−3.63	−3.63	−3.63	−3.63	−3.63
Rec09	−2.38	−4.48	−3.60	−3.60	−4.62	−4.58	−4.58	−4.62	−4.62	−4.62
Rec11	−1.54	−3.34	−1.44	−1.44	−3.34	−3.34	−3.34	−3.34	−3.34	−3.34
Rec13	−5.76	−5.65	−3.43	−3.43	−6.05	−6.05	−6.05	−6.05	−6.05	−6.05
Rec15	−5.91	−6.02	−4.83	−4.83	−5.91	−6.02	−5.91	−5.93	−6.02	−6.02
Rec17	−5.15	−5.47	−5.15	−5.15	−5.58	−5.58	−5.58	−5.58	−5.58	−5.58
Rec19	−7.57	−5.45	−7.70	−7.44	−9.72	−9.25	−9.38	−9.72	−9.72	−9.72
Rec21	−4.21	−2.22	−3.68	−3.68	−6.31	−6.30	−6.17	−6.23	−6.43	−6.43
Rec23	−10.8	−6.70	−7.29	−7.29	−10.8	−10.7	−10.89	−10.9	−10.9	−10.9
Rec25	−5.45	−2.69	−3.08	−3.08	−5.97	−6.31	−6.21	−6.17	−6.31	−6.31
Rec27	−5.83	−2.60	−3.64	−3.64	−5.64	−6.10	−5.83	−6.10	−6.13	−6.13
Rec29	−7.23	−3.99	−7.23	−7.23	−7.94	−8.28	−7.94	−8.02	−8.15	−8.15
Rec31	−4.71	2.72	−3.76	−3.78	−5.90	−6.13	−6.22	−6.68	−6.74	−6.88
Rec33	−5.35	4.78	−1.97	−2.01	−5.51	−6.31	−6.37	−6.79	−6.79	−7.01
Rec35	−5.51	3.67	−4.94	−4.94	−6.08	−6.17	−5.91	−6.65	−6.80	−6.72
Rec37	−10.0	5.89	−7.80	−7.92	−9.41	−9.49	−9.36	−10.4	−10.4	−10.4
Rec39	−5.32	8.80	−4.97	−5.12	−7.00	−6.99	−6.91	−7.53	−7.71	−7.56
Rec41	−7.41	6.79	−6.08	−6.08	−8.78	−8.57	−8.82	−9.26	−9.32	−9.28

program the QDEA and apply it to no-wait PFSP. The values of BRE for all these approaches are shown in Table 2 along with that of HQEA.

From Table 2, it can be seen that the BRE values obtained from HQEA are better than those of VNS, GASA, DS, DS+M, TS, TS+M and TS+MP almost for all instances. Compared to the DPSO which is one of the most effective algorithms for the no-wait FSP, our HQEA performs approximately in the same level with DPSO since most of BRE for these two algorithms are exactly the same. All these demonstrate the effectiveness of the proposed HQEA for no-wait PFSP. Meanwhile, for the QDEA, although the differential operation is used to perform the quantum updating which works like the way we adopted in this paper, the sorting based decoding strategy used for solution representation and the variable neighborhood search (VNS) performed on the permutative job sequence make it different with our HQEA. Compared to the QDEA, our HQEA also can obtain better or at least equal scheduling results as shown in Table 2. So we can conclude that the decoding strategy and the local search proposed in this paper are suitable for dealing with the no-wait PFSP and can be applied to tackle these types of optimization problems.

4 Conclusions

In this paper, we proposed a novel hybrid quantum-inspired evolutionary algorithm for solving the no-wait permutation flow shop scheduling problems. Based on the QEA, we studied the application of the HQEA by adopting a novel strategy to perform the updating of quantum gate and an effective converting mechanism for determining the solution for no-wait PFSP. We made the simulations based on famous benchmark problems and the results demonstrated the effectiveness of HQEA. For the future work, we can adopt this approach to deal with other shop scheduling problems like jobshop scheduling problems and make the comparisons with other algorithms.

References

1. Rock, H.: The three-machine no-wait flowshop problem is NP-complete. J. Assoc. Comput. Machinery 31, 336–345 (1984)
2. Rajendran, C.: A no-wait flowshop scheduling heuristic to minimize makespan. J. Oper. Res. Soc. 45(4), 472–478 (1994)
3. Laha, D., Chakraborty, U.K.: A constructive heuristic for minimizing makespan in no-wait flow shop scheduling. International Journal of Advanced Manufacturing Technology 41, 97–109 (2009)
4. Aldowaisan, T., Allahverdi, A.: New heuristics for no-wait flowshops to minimize makespan. Computers and Operations Research 30, 1219–1231 (2003)
5. Schuster, C.J., Framinan, J.M.: Approximative procedures for no-wait job shop scheduling. Oper. Res. Lett. 31(3), 308–318 (2003)
6. Grabowski, J., Pempera, J.: Some local search algorithms for no-wait flow-shop problem with makespan criterion. Computers and Operations Research 32, 2197–2212 (2005)
7. Liu, B., Wang, L., Jin, Y.-H.: An effective hybrid particle swarm optimization for no-wait flow shop scheduling. International Journal of Advanced Manufacturing Technology 31, 1001–1011 (2007)
8. Pan, Q.-K., Tasgetiren, M.F., Liang, Y.-C.: A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Computers & Operations Research 35, 2807–2839 (2008)
9. Han, K.-H., Kim, J.-H.: Quantum-inspired Evolutionary Algorithm for a Class of Combinatorial Optimization. IEEE Trans on Evolutionary Computation (2002)
10. Wang, L., Wu, H., Tang, F., Zheng, D.Z.: A hybrid quantum-inspired genetic algorithm for flow shop scheduling. In: Huang, D.-S., Zhang, X.-P., Huang, G.-B. (eds.) ICIC 2005. LNCS, vol. 3645, pp. 636–644. Springer, Heidelberg (2005)
11. Gu, J., Gu, X., Jiao, B.: A Quantum Genetic Based Scheduling Algorithm for stochastic flow shop scheduling problem with random breakdown. In: Proceedings of the 17th World Congress. The International Federation of Automatic Control Seoul, Korea, July 6-11, pp. 63–68 (2008)
12. Niu, Q., Zhou, T., Ma, S.: A Quantum-Inspired Immune Algorithm for Hybrid Flow Shop with Makespan Criterion. Journal of Universal Computer Science 15, 765–785 (2009)

13. Zheng, T., Yamashiro, M.: Solving flow shop scheduling problems by quantum differential evolutionary algorithm. The International Journal of Advanced Manufacturing Technology 49(5-8), 643–662 (2010)
14. Storn, R., Price, K.: Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report, TR-95-012, ICSI (1999)
15. Carlier, J.: Ordonnancements a Contraintes Disjonctives. Recherche Operationelle /Operations Research 12, 333–350 (1978)
16. Reeves, C.R.: A genetic Algorithm for Flowshop Sequencing. Computers and Operations Research 22, 5–13 (1995)

Reducing the Search Space in Evolutive Design of ARIMA and ANN Models for Time Series Prediction

Juan J. Flores[1], Hector Rodriguez[1], and Mario Graff[2]

[1] Division de Estudios de Postgrado
Facultad de Ingenieria Electrica
Universidad Michoacana de San Nicolas de Hidalgo, Mexico
[2] School of Computer Science and Electronic Engineering
University of Essex, UK
juanf@umich.mx, hector120713@gmail.com, mgraff@eseex.ac.uk

Abstract. Evolutionary design of time series predictors is a field that has been explored for several years now. The levels of design vary in the many works reported in the field. We decided to perform a complete design and training of ARIMA models using Evolutionary Computation. This decision leads to high dimensional search spaces, whose size increases exponentially with dimensionality. In order to reduce the size of those search spaces we propose a method that performs a preliminary statistical analysis of the inputs involved in the model design and their impact on quality of results; as a result of the statistical analysis, we eliminate inputs that are irrelevant for the prediction task. The proposed methodology proves to be effective and efficient, given that the results increase in accuracy and the computing time required to produce the predictors decreases.

Keywords: Evolutionary Computation, Artificial Neural Networks, ARIMA models, Time Series Forecasting.

1 Introduction

A time series is a sequence of observations of a given variable at equally spaced periods of time. Time series result from the observation of processes in a bast range of application areas such as: finance, economics and wind speed, among others. Most of the systems that produce the time series can be modelled as dynamic systems. In some cases, we do not have the domain knowledge of the relationships between variables and therefore we cannot establish the differential equations that govern the system of interest. Even in those cases, we can observe the system's variables and produce time series out of those observations.

One of the most widely used models to forecast behavior of time series is the Box-Jenkins approach, also known as ARIMA models. This family of models predict the behavior of the system by forming a function of past observations and errors produced by the same model (see Section 3). ARIMA models have been traditionally produced by statistical means, although, recently evolutionary computation has been used to determine the best model for the forecasting task.

Another, also well known, alternative to solve the time series forecasting problem is to perform the statistical analysis to determine the inputs to the ARIMA model, and substitute the ARIMA model by an Artificial Neural Network (ANN) (e.g. [1]).

ARIMA models are linear models, while ANN are non-linear models. ANN have been shown to be universal approximators [11]. So, an ANN model that considers the same inputs as an ARIMA model is expected to have a smaller error; i.e. ANNs are better approximators than linear models.

In producing ARIMA models or ARIMA-like ANN, the goal is to obtain the model that minimizes the forecasting error. Obtaining the best model is, thus, an optimization problem. Some works assume a given ANN architecture and use evolutionary computation to train the network. Others design the architecture and use back-propagation to train the network. Recently, Flores *et al.* ([8]) have used evolutionary computation to solve both problems, the design of the ANN architecture, and the training of the network.

In any case, the search spaces to determine ARIMA or ARIMA-like ANN models are very large. The size of the search space grows linearly with the number of variables for ARIMA models, and quadratically with the number of variables for ANN models. This growth impacts exponentially the size of the search spaces. That is, the size of the populations for any evolutionary algorithm has to grow exponentially with the dimensionality of the problem, in order to maintain a constant density of exploration of the search space.

Searching such a space, with an exponentially growing number of individuals, means that the number of evaluations, and therefore the required time to produce an optimal forecaster grows exponentially with the number of variables involved. This effect is known as the *course of dimensionality*. See [20] for more details.

If we want to produce effective and efficient forecasters, we require to perform a more intelligent search in those large search spaces. In particular, through a preliminary skimming of the search space, we can gather information that lead us to prune it. Pruning the search space allow us to search only those regions that look more promising. The result of the pruning process is the production of better forecasting models in less time.

The rest of the paper is organized as follows. Section 2 presents the related work. There traditional features selection method and its limitation are mentioned as well as the previous work where ANN are used as predictors. Section 3 describes the ARIMA models and ANN in the context of time series prediction. Section 4 presents our methodology to reduce the size of the search space. The methodology used to evolve the models and the results are shown in Sections 5 and 6, respectively. Some conclusions and possible directions for future work are given in Section 7.

2 Related Work

Wind forecasting techniques assume that the time series, taken from measurements, is the sum of different components and a random error. The goal of most forecasting techniques is to separate and identify those components (trend, cyclical, seasonal, and irregular). Recently, several techniques have been used from the fields of statistics and artificial intelligence [10,3,1,18]. Scientists have even combined them in order to reduce the forecasting error and to produce more accurate predictions [22,23].

There have been a number of studies of wind speed behavior at La Venta, Oaxaca [6,19,12]. Regarding time series forecasting, Cadenas and Rivera [1,2], and Flores *et al.* [7], have made models for this purpose. Cadenas and Rivera [1] discuss ARIMA techniques and Artificial Neural Networks (ANN) and make a comparison between the two through the calculation of statistical errors (MAE, MSE, and Theil's U). The final result shows that ARIMA is the best for this case; however, the authors mentioned that the factor that limits ANN performance is the size of the training set presented to the network. Cadenas and Rivera [2] present a comparison of ANN with different configurations, under minimum operating requirements, and suggested the network model with the minimum statistical errors. Flores *et al.* [7] modelled wind speed with genetic programming, producing a forecasting model which reduces the statistical errors generated by the ARIMA technique. Flores *et al.* [8] also modelled the wind speed using ANN with evolutionary computation showing more accuracy compared against models generated with genetic programming and the models proposed by Cadenas. But in order to achieve this better accuracy more computational time was required. This was cause by the dimensionality of the search space.

Moving away from the ANN field, our work is also related to feature selection that uses methods such as [5,9]. These methods choose the predictors based on the correlation between them and the output of the system. However, traditionally these methods can only include the lag variables and are inapplicable to select the moving average predictors of the ARIMA model.

The problem of feature selection has also been tackled using evolutionary techniques. In [13,14] a two stage genetic programming was used to select the features. That is, in the first stage GP used all the possible features to make classifiers. This process was repeated n times thus creating n best-of-run classifiers. Then the process was repeated but now with only those features which were used in 10 or more of the classifiers. At this stage all the features were used at least once; however, there were features that appeared more frequently than others and then those "popular" features were selected to create the final model. These works used a similar approach than the one proposed here. However, our procedure presents significant differences with respect to them.

Firstly, we focus on artificial neural networks. To the best of our knowledge, no similar procedure has ever been applied to evolve ANN. Secondly, we not only select the features based on the frequency they appear in the evolved models, but also we take into consideration the quality of the ANN solution in which those features appeared. Thirdly, we present a quality comparison between the

ANN evolved using our procedure and ANNs evolved using all the features. This comparison is presented in terms of the quality of the predictions and the time required to obtain the ANN.

Our work bear some similarities with the methodology proposed in [16] and [4]. There a GA was used to evolve an ARMA and ARIMA models, respectively. The GA was used to obtain the order of the autoregressive and moving average components as well as the different variables involved in the model. As can be seen, this is similar to objective of our work; however, there are some significant differences. Firstly, here we use also a evolutionary algorithm to estimate the model coefficients. Secondly, we propose a procedure to reduce the search space that not only take in consideration the number of times a particular variable appear in the model but also the quality of the model in which they appear.

Stochastic algorithms have also been proposed as an alternative to traditional approaches for the estimation of model's coefficients. In [21] a PSO was used to do this job and a statistical approach identified the structure of the ARIMA model.

3 ARIMA and ANN Models

An ARIMA model [15] is a statistical model that allows us to model time series, and to predict their behavior. These models have the following form:

$$\begin{aligned} y_t &= \sum_{k=1}^{w} a_k y_{t-k} + \sum_{k=1}^{w} b_k e_{t-k} + \epsilon_t \\ e_t &= y_t - \hat{y}_t, \end{aligned} \tag{1}$$

where y_t represents the measurement at time t in the time series and $\hat{y}_t$ is the forecasting produced by the ARIMA model; ϵ_t represents the effects of random factors; w is the window width. The window represents how far behind in time we consider measurements as probably important inputs for the ARIMA model. Outside of the window, observations are not taken into account. Using statistical procedures, the numerical value of the coefficients a_k and b_k are determined.

The ANN architecture used for prediction is the Multi-Layer Perceptron (MLP). A MLP, as a universal approximator [11], can learn any function, given it has enough neurons in the hidden layer. That fact allows the network to capture the different forms of the function to be modeled. Given an AR model, we can design a MLP capable of reproducing the time series at least as well as the ARIMA model itself. The output of the MLP is always a single neuron, representing the forecasting output, $\hat{y}_t$. Once the inputs to the MLP are specified, the design process reduces to determine the number of neurons in the hidden layer. Notice that the learning models for ANNs are designed to determine the weights of the synaptic connections. Those learning models do not consider the design of the network architecture. One way to design the neural network is to perform a statistical analysis to determine what variables are important in the forecasting. Those variables will be considered as the inputs to the ANN.

4 Reducing the Search Space

It is a well know phenomena in the field of GA and Genetic Programming that some of the variables or building components tend to disappear during the evolutionary process. Inspired by this behaviour we started to analyse whether this behaviour is present in the problem of evolving ANN predictors. Our preliminary experiments shows us that the frequency of some of the lag variables decreases to almost zero at the end of the evolutionary process while other variables become dominant. In fact, this behaviour can be explain by Prince's theorem [17] which shows that the number of fit genes will increase each generation while the number of unfit genes will decrease. This suggests that it may be possible to use an evolutionary algorithm to discover which variables are important.

To prove this, we used a GA to train an ANN and to instantiate an ARIMA model. The GA uses all the possible lag variables and the error terms up to a limit, which for this time series was set to 18 (i.e., $y_{t-1}, \ldots, y_{t-18}, e_{t-1}, \ldots, e_{t-18}$), run for a number of generations and tries to find the model that makes the fittest prediction in a training set. From these characteristics, GA proceeds as it normally does in any other application. Being the only difference that we store for each individual evaluated (i.e., model) its fitness and the lag variables and error terms it uses.

From the information stored, we can infer not only which of the terms are more frequently used but also the quality of the models in which they appear. In order to decide which of the terms (i.e., y_{t-i} and e_{t-i}) to use we decided to compute the weighted frequency for each one of the involved variables. That is,

$$h(y_{t-i}) = \frac{1}{a} \cdot \sum_{j \in \mathcal{G}(y_{t-i})} \frac{1}{f(j)} \tag{2}$$

$$a = \sum_{i=1}^{w} \sum_{j \in \mathcal{G}(y_{t-i})} \frac{1}{f(j)} + \sum_{j \in \mathcal{G}(e_{t-i})} \frac{1}{f(j)},$$

where $\mathcal{G}(x)$ is the set composed by the models where x (i.e., y_{t-i} or e_{t-i}) was involved and $f(j)$ is the fitness of model j.

Fig. 1 presents the weighted frequency of all the ARIMA terms that corresponds to the best ANN model obtained for the experiments presented in Section 6. The first 18 bins are the y_{t-i} terms and the rest 18 bins are the e_{t-i}. As can been seen, the figure can be used to discriminate which of the variables are more "important" according to the fitness of the models produced and the frequency of the variables. For example, our procedure selected the term y_{t-1} to be the most important variable, since it is the variable that appears in more good models.

There are a number of procedures one could use to decide which variables to include in the final model based on the information depicted in Fig. 1. For example, one could decide to include the n variables presenting the higher values or compute some statistical measure on those values and decide based on that measure. In our proposal we are including those variables that have a frequency

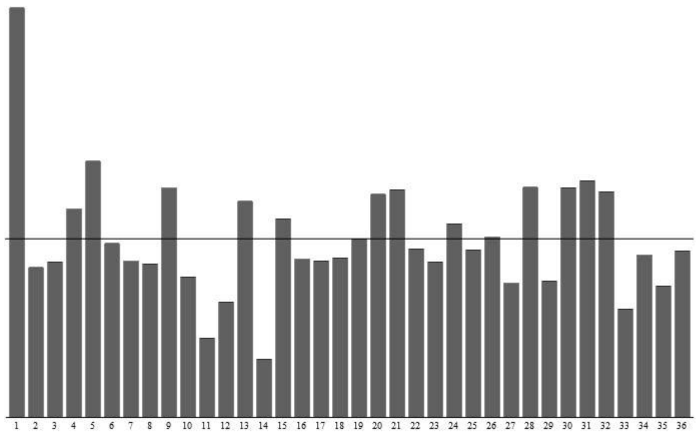


Fig. 1. Weighted frequency of the ARIMA variables of the best model of the ANN experiments. The first 18 bins corresponds to the terms y_{t-1} and the rest to the terms e_{t-i}. The horizontal line represents the mean of these values.

value greater than the mean. That is, we average the values of all the terms and include in the final models only those variables that have a value higher than the mean. The mean is shown in Fig. 1 with a horizontal line.

In the example shown in Fig. 1, only 14 out of 36 variables have a weighed frequency higher than the average. The search space is being pruned considerably; its dimensionality has decreased from 36 to 14. This fact speeds up the search process; furthermore, by considering only those variables that appear in promising models, we are discarding the inclusion of not so important variables, spending the computing time in the analysis of the best models. Section 6 presents the results obtained using this procedure.

5 Evolving ARIMA Models and ANN

The process used to obtain the ARIMA models and to train the ANN is a two-step evolutionary technique. In the first step, the data needed by the variable selection procedure, described in Section 4, is collected and, thus, at the end of it we have the set of variables used in the final model. Now that the variables involved in the model have been identified, the next step is to estimate the coefficients of the ARIMA models or the weights of the ANN. In all cases, we used a generational genetic algorithm (GA) with fitness proportionate selection, one-point crossover (70%), and uniform mutation (70% with 20% of point mutation).

The variables are identified using two nested GA processes. The individuals of the first process (outer loop) contains the possible variables to use in the models and the number of hidden neurons for the case of ANN. That is, the outer loop proposes the architecture of the model. The second evolutionary process (inner loop) estimates the coefficients of the ARIMA model or the weights of the ANN, depending of the model being evolved. That is, the inner loop instantiates the ARIMA model or trains the ANN. At the end of this process the fitness of the evolved model is computed in the training set. This fitness is then used in the outer loop to guide the evolutionary process.

In the outer loop we used two different structures to represent the architecture of the model. For the case of ARIMA models we used a binary string of length 36 that represents which variables are active in the model. Fig. 2 depicts the structure used for this case.

y_{t-1}	...	y_{t-18}	e_{t-1}	...	e_{t-18}
0	...	1	1	...	1

Fig. 2. Chromosome used in the ARIMA experiments to decide which variables to include in the model

For the case of evolving an ANN, besides the inputs to the net we require the number of neurons in the hidden layer. As a result, we divided the chromosome in two parts. The first part is equivalent to the one used in the ARIMA models and the second part encodes the number of hidden neurons into a binary string. The minimum number of neurons contained in the hidden layer is set to 18. As a result, we are setting a limit to the number of neurons of the hidden layer. Fig. 3 shows the structure used for the ANN. The first part represents the variables used as inputs in the ANN and the second part (shaded in the figure) indicates the number of neurons in the hidden layer.

y_{t-1}	...	y_{t-18}	e_{t-1}	...	e_{t-18}	n
0	...	1	1	...	1	1 0 1 0 1

Fig. 3. Chromosome used in the ANN experiments to decide which variables to include in the model. The last part of the chromosome (shaded in part) represents the number of hidden neurons of the net.

The inner loop in both cases uses a real coded GA in order to identify the coefficients of the ARIMA model or the weights of the ANN. The only difference is the length of the chromosome. For instance, in the ARIMA case the length of the chromosome is specified by the number of variables involved in the model. On the other hand, in the ANN experiments the length of the chromosome is specified by the number of variables involved and the number of hidden neurons. That is, let m be the number of variables and n the number of hidden neurons. Then, the length of the chromosome is $n \times m + n$.

The ANN we decided to use is a fully connected feed-forward network with 3 layers and n hidden neurons. We decided to use a fully connected feed forward network because it is the best understood architecture, it has been proven to be a universal approximator and has been used in previous works to predict times series. The activation function was a sigmoid in all neurons.

Table 1. Number of individuals evaluated for each step in the proposed methodology and the traditional procedure that does not use the reduction of the search space

	Reduction of Search Space			Full Search Space	
Model	Architecture identification Outer loop	Inner loop	Parameter Estimation	Outer loop	Inner loop
ARIMA	650	650	160,700	10,250	1,100
ANN	330	330	16,300	2,040	1,400

For comparison proposes, we performed another series of experiments which does not perform the reduction of the search space. That is, it uses only the first step of the procedure described here which is the two nested GA processes. In this process, the final model is the best model found. In order to make a fair comparison between our approach and the models evolved with the full search space, we decided to use more computational to the later procedure. Tab. 1 shows the number of individuals evaluated for each methodology. As can be seen from the table, we evaluated more individuals in the procedure that uses the full search space than in the methodology proposed here.

6 Results

Let us start this section describing the wind time series used to test our methodology. The Comisión Federal de Electricidad (CFE, the governmental electricity supplier in México) has made wind speed measurements since 1994, through a network of measurement stations located in places of interest. The sensors are located at different heights in the measurement towers (20m, 30m, and 40m from ground level). Sensor's characteristics are shown in Tab. 2.

Fig. 4 shows the monthly behavior of the wind speed in La Venta, Oaxaca, for the period June 1994 to May 2000. From the figure, it may be observed a seasonal behavior as well as the tendency of having the stronger wind speed at the end of the year and in the middle of the year the wind speed is more calm.

As mentioned previously, we performed two sets of experiments. One to obtain the ARIMA model and the other for the ANN predictor. Furthermore, for the sake of comparison we evolved ARIMA and ANN models using a reduced search space; we also included an ARIMA model created using the Box-Jenkins approach (presented in [1]).

Table 2. Specification of the measurement sensors

Specification	Anemometer	Wind Vane
Measuring rank	0.78-45 m/s	0-360
Exactness	5	5
Resolution	0.78m/s	1 m/s

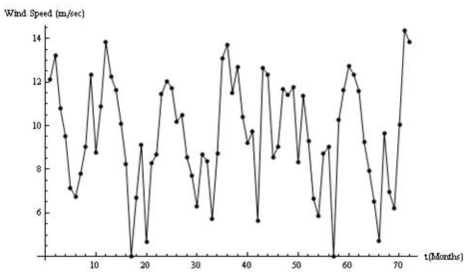


Fig. 4. Wind speed times series in La Venta, Oaxaca

For the case of ARIMA models. Tab. 3 shows the variables that were selected by the procedure described in Section 4. The complete ARIMA model is shown in Equation (3).

$$\begin{aligned} y_t = & \ 0.4y_{t-1} + 0.23y_{t-2} + 0.68y_{t-3} - 0.3y_{t-4} + 0.2y_{t-9} \\ & -0.29y_{t-10} + 0.76y_{t-12} - 0.48y_{t-14} + 0.28y_{t-15} \\ & +0.03y_{t-16} - 0.2e_{t-3} + 0.006e_{t-5} + 0.1e_{t-7} + 0.38e_{t-11} \\ & -0.36e_{t-12} + 0.002e_{t-13} + 0.13e_{t-17} - 0.16e_{t-18}. \end{aligned} \tag{3}$$

For the case of ANN, Tab. 4 shows the variables that were selected as inputs in the final step of our methodology.

In order to analyze whether our methodology is able to select the variables that one would select using a traditional approach such as Box-Jenkins, Equation (4) shows the ARIMA model proposed in [1]. As can be seen, comparing Tables 3 and 4 and Equation (4), our procedure selected almost all variables included using a traditional approach. Furthermore, it incorporated variables that were not selected by the statistics. We will see that these variables neglected by the statistics are necessary to obtain a model with better accuracy.

$$\begin{aligned} y_t = & \ t_{t-1} + y_{t-12} - t_{t-13} - 0.997e_{t-1} \\ & - 0.7976e_{t-12} + 0.7956e_{t-13}. \end{aligned} \tag{4}$$

In order to have a complete picture of the quality of the models produce with our methodology, Tab. 5 shows the accuracy of the models in the validation set in

Table 3. Selected variables for the ARMA model

Inputs	y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}, y_{t-9}, y_{t-10}, y_{t-12}, y_{t-14}, y_{t-15},y_{t-16}
Errors	e_{t-3}, e_{t-5}, e_{t-7}, e_{t-11}, e_{t-12}, e_{t-12}, e_{t-13}, e_{t-17}, e_{t-18}

Table 4. Selected variables for the ANN model

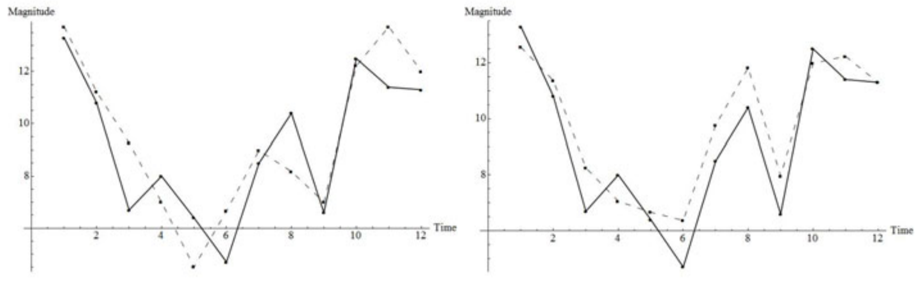
Inputs	y_{t-1}, y_{t-4}, y_{t-5}, y_{t-9}, y_{t-13}, y_{t-15}
Errors	e_{t-2}, e_{t-3}, e_{t-6}, e_{t-8}, e_{t-10}, e_{t-12}, e_{t-13}, e_{t-14}

terms of the mean square error (MSE). The table also includes the time needed to create the models. As can been seen from the table, the lowest MSE values and lowest time correspond to the ANN model obtained with the method to reduce the search space (pruning). In fact, comparing the MSE values and time of the models obtained with the traditional procedure and the reduced search space, we can see that in both cases the models evolved in a reduced search space exhibit better accuracy and furthermore the time needed to obtain them is shorter. The last column shows the accuracy of the ARIMA model obtained using the statistical approach. These values corroborate that, for this time series, our procedure not only selects the variables that are "important" but also makes better predictions.

Table 5. Comparison in terms of the mean square error in the validation of the ARIMA models, the ANN using GA with the traditional procedure and with reduced search space. The last column presents an ARIMA model using a statistical approach.

	ARIMA (GA)		ANN		ARIMA
	Normal	Pruning	Normal	Pruning	
Training set	4.53	3.79	3.41	**0.87**	4.43
Validation set	5.69	2.22	3.36	**0.90**	3.68
Time (minutes)	63.75	38.3	38.1	**26.1**	N/A

While the MSE values provides an objective indicator of the quality of the models produced. It may be hard for the reader to appreciate the accuracy of the predictions based on such figures. In order to provide a more visual indication of the accuracy of the models, Fig. 5 shows the predictions in the validation set (dotted line) for the ARIMA model and the ANN. The solid line in the figures presents the actual time series. As can be seen both models follows closely the actual time series.



(a) ARIMA (Validation Set) (b) ANN (Validation Set)

Fig. 5. Predictions made in the validation set (dotted line) with the best ARIMA model and the ANN. The solid line represents the actual time series.

7 Conclusions

We presented a methodology to reduce the search space of ARIMA and ANN predictors. The idea here is to collect information from an evolutionary process and decide, based on that information, which of the variables are the most important. By doing so, we are effectively reducing the dimensionality of the search space and, as a consequence, its size.

In order to test our approach, we decided to create predictors following the structure of an ARIMA model and ANN. The difference between our methodology and previous work is that our methodology uses GA to identified both the architecture of the model, as well as to estimate the coefficients of the ARIMA model or weights of the ANN. We compare our methodology with a similar procedure, which that does not use the reduction of the search space and with an ARIMA model identified using the Box-Jenkins approach. In all cases our methodology produced better models in terms of accuracy and also consumed less computational resources.

Finally, we would like to suggest possible future work. We tested our methodology with the wind time series, as a future work we would like to test this approach with time series involving more variables. Furthermore, the procedure used to select the variables makes the decision based on the mean, we would like to test whether another statistical test produce better results or going even further by selecting only the m best variables.

References

1. Cadenas, E., Rivera, W.: Wind speed forecasting in the south coast of Oaxaca, Mexico. Renewable Energy 32(12), 2116–2128 (2007)
2. Cadenas, E., Rivera, W.: Short term wind speed forecasting in La Venta, Oaxaca, using artificial neural networks. Renewable Energy 34, 274–278 (2009)
3. Chen, Y., Yang, B., Dong, J., Abraham, A.: Time-series forecasting using flexible neural tree model. Information Sciences 174, 219–2352 (2005)
4. Chorng-Shyong, O., Jih-Jeng, H., Gwo-Hshiung, T.: Model identification of ARIMA family using genetic algorithms. Applied Mathematics and Computation 164(3), 885–912 (2005)
5. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. Annals of Statistics 32(2), 407–499 (2004)
6. Elliot, D., Schwartz, M., Scott, G., Haymes, S., George, R.: Wind energy resource atlas of Oaxaca. NREL/TP, 500-34519 (2003)
7. Flores, J.J., Graff, M., Cadena, E.: Wind prediction using genetic programming and gene expression programming. In: Techniques and Methodologies for Modelling and Simulation of Systems, Lyon France - Mexico, AMSE, International Association for Advanced of Modelling and Simulation, pp. 34–40 (2005) ISBN:970-703-323-1
8. Flores, J.J., Loaeza, R., Rodriguez, H., Cadenas, E.: Wind speed forecasting using a hybrid neural-evolutive approach. In: Aguirre, A.H., Borja, R.M., García, C.A.R. (eds.) MICAI 2009. LNCS, vol. 5845, pp. 600–609. Springer, Heidelberg (2009)
9. Gelper, S., Croux, C.: Least angle regression for time series forecasting with many predictors. Open Access publications from Katholieke Universiteit Leuven urn:hdl:123456789/164224, Katholieke Universiteit Leuven (2008)

10. Ghiassi, H., Saidane, M., Zimbra, D.K.: A dynamic artificial neural network model for forecasting time series events. International Jounal of Forecasting 21, 341–362 (2005)
11. Haykin, S.: Neural Networks a comprehensive foundation. Prentice-Hall, Englewood Cliffs (1999)
12. Jaramillo, O.A., Borja, M.A.: Wind speed analysis in La Ventosa Mexico: a bimodal probability distribution case. Renewable Energy 29, 1613–1630 (2004)
13. Langdon, W.B., Buxton, B.F.: Genetic programming for mining DNA chip data from cancer patients. Genetic Programming and Evolvable Machines 5(3), 251–257 (2004)
14. Langdon, W.B., Harrison, A.P.: GP on SPMD parallel graphics hardware for mega bioinformatics data mining. Soft Comput. 12(12), 1169–1183 (2008)
15. Makridakis, S., Whellwright, S.C., Hyndamn, R.J.: Forecasting Methods and Applications, 3rd edn. John Wiley & Sons, Inc., Chichester (1992)
16. Minerva, T., Poli, I.: Building ARMA models with genetic algorithms. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 335–343. Springer, Heidelberg (2001)
17. Price, G.R.: Selection and covariance. Nature 227, 520–521 (1970)
18. Riahy, G.H., Abedi, M.: Short term wind speed forcasting for wind turbine applications using linear prediction method. Renewable Energy 33, 35–41 (2008)
19. Steenburgh, W., Schultz, D., Colle, B.: The structure and evolution of gap outflow over the Gulf of Tehuantepec. Monthly Weather review 91 (1998)
20. Verleysen, M., François, D.: The curse of dimensionality in data mining and time series prediction. In: Cabestany, J., Prieto, A.G., Sandoval, F. (eds.) IWANN 2005. LNCS, vol. 3512, pp. 758–770. Springer, Heidelberg (2005)
21. Wang, H., Zhao, W.: Arima model estimated by particle swarm optimization algorithm for consumer price index forecasting. In: Deng, H., Wang, L., Wang, F.L., Lei, J. (eds.) AICI 2009. LNCS, vol. 5855, pp. 48–58. Springer, Heidelberg (2009)
22. Zhang, G.P.: Time series forecasting using a hybrid arima and neural network model. Neurocomputing 50, 159–175 (2003)
23. Zhang, G.P.: A neural network ensemble method with jittered training data for time series forecasting. Information Sciences 177, 5329–5346 (2007)

Routing Algorithms for Wireless Sensor Networks Using Ant Colony Optimization

Christian Domínguez-Medina and Nareli Cruz-Cortés

Center for Computing Research,
National Polytechnic Institute (CIC-IPN), Mexico
chrisdom27@hotmail.com, nareli@cic.ipn.mx

Abstract. Wireless Sensor Networks have become an active research topic in the last years. The routing problem is a very important part in this kind of networks that need to be considered in order to maximize the network life time. As the size of the network increases, routing becomes more complex due the amount of sensor nodes in the network. Sensor nodes in Wireless Sensor Networks are very constrained in memory capabilities, processing power and batteries. Ant Colony Optimization based routing algorithms have been proposed to solve the routing problem trying to deal with these constrains. We present a comparison of two Ant Colony-based routing algorithms, taking into account current amounts of energy consumption under different scenarios and reporting the usual metrics for routing in wireless sensor networks.

Keywords: Wireless Sensor Networks (WSN), network life time, routing algorithms, Ant Colony Optimization.

1 Introduction

Wireless Sensor Networks (WSN) consist of a large number of embedded sensors having the capability to communicate among them via wireless links deployed in an area that should be monitored. WSN are very effective in many fields such as intrusion detection, weather monitoring, security and tactical surveillance, distributed computing, detecting ambient conditions such as temperature, movement, sound, light, or the presence of certain objects, inventory control, and disaster management [8]. Nowadays, these sensor nodes are equipped with a small processor, constrained memory and constrained wireless communication capabilities. Furthermore, the sensor nodes are very limited in terms of their battery capabilities. For this reason, it is crucial to handle their energies properly [6]. A node in the WSN measures some phenomena from the environment, then it sends the measured data through others nodes to the base station. This base station could be connected to an application or the Internet. The strategy to build the path to the base station is known as a routing algorithm. The algorithms inspired by some biological phenomena have become popular in some Artificial Intelligence communities mainly because they have demonstrated to be competitive options to solve some hard problems from engineering and science. Specifically, a family of

Ant Colony Optimization (ACO) algorithms [4] have been successfully applied to solve some routing problems in wired and wireless networks. We selected two ant-based routing algorithms and programmed and executed them in order to compare their performance under some metrics. The main contribution of this paper is the calculation of these metric values based on realistic amounts of energy consumption. Our goal is to give some light on the real earnings and feasibility of using this kind of algorithms. The remainder of the paper is organized as follows. In Section 2 the related work is presented. Section 3 describes the ACO algorithms. The Experiments are explained in Section 4. The results are described in Section 5, and finally in Section 6 some conclusions are drawn.

2 Related Work

WSN can be considered ad-hoc networks. However, protocols for Mobile Ad-hoc Networks (MANETs) can not be successfully applied to them because of WSN's special features [2]. There are a wide range of routing protocols that have been used to solve the problem of routing in WSN, for example a hierarchical clustering algorithm for sensor networks, called "Low Energy Adaptive Clustering Hierarchy" (LEACH) [7] and the protocol called "Power-Efficient Gathering in Sensor Information Systems" (PEGASIS) [10], they are two of the most used routing protocols. LEACH is a cluster-based protocol, which includes distributed cluster formation. It randomly selects a few sensor nodes as clusterheads and rotate this role to evenly distribute the energy load among the sensors in the network. The basic idea of the protocol PEGASIS is that in order to extend network lifetime, nodes need only communicate with their closest neighbors and they take turns in communicating with the base station. When the round of all nodes communicating with the base station ends, a new round will start, and so on. This reduces the power required to transmit data per round as the power draining is spread uniformly over all nodes.

Moreover, very recently, some researchers have proposed routing protocols based on the ACO algorithm, among them we can mention the following:

In [1] a Quality of Service (QoS) routing solution is proposed, it is named ACO based QoS routing algorithm (ACO-QoSR), it searches for the best paths that satisfied the QoS requirements by using intelligent artificial ants. ACO-QoSR algorithm is the tradeoff between a certain guaranteed QoS requirements and acceptable computational complexity.

In [17] is proposed a pheromone based energy-aware directed diffusion algorithm (PEADD) for WSN that extend the network lifetime by using pheromone and enhances the network reliability by maintaining remaining energy distribution relatively uniform among sensor nodes.

In [5] are shown the properties and review the main instances of network routing algorithms whose bottom-up design has been inspired by collective behaviors of social insects such as ants and bees.

In [12] authors proposed an algorithm based on ACO for flat architectures and localization awareness. This proposal tries to maximize the network lifetime and deal, react and adapt itself to changes in the network.

In [15] an Energy Delay Based on ACO (E&D) whose main goal is to find the optimal routing not only to maximize the lifetime of the network but also to provide real-time data transmission services.

In [14] the authors proposed the called Ant Colony Optimization-Based Location-Aware Routing (ACLR) which is a flat and location awareness algorithm. It fuses the residual energy and the global and local location information of nodes, to define the probability to select the next hop for the ants.

In [2] is proposed an Energy-Efficient Ant-Based Routing Algorithm (EEABR) for flat and location awareness architectures. In this proposal, the ants look for less energy consuming paths meanwhile reducing the size of the ants during the communication among nodes. Other similar ACO-based routing algorithms for WSN are published in [9] where authors shown crucial biologically inspired mechanisms and the associated techniques for resolving routing in WSN, including ant based and genetic approaches.

In [11] a novel routing approach using an ACO algorithm is proposed for WSN consisting of stable or limited mobile nodes. This approach is also implemented to a small sized hardware component as a router chip.

In [16] a routing algorithm for data aggregation based on ACO (ACAR) is presented. The main idea of this algorithm is optimization of data aggregation route by some cooperation agents called ants using the three heuristic factors about energy, distant and aggregation gain.

Dorigo and Di Caro show a method in [3], which is called Ant-Net, based on ant's colonies. In this method the information that the ants provide appears in each node like a routing table and a data structure that involve information about local traffic and delay quantities. Below presents the basic ACO algorithm and the two algorithms we are going to compare.

3 Ant Colony Optimization-Based Routing Algorithms for WSN

The main characteristic of an ACO routing algorithm consists in the continual acquisition of routing information through path sampling by using small control packages called *ants*. The ants are placed inicially in the source node s_o with the task of find out paths through the other nodes to the destination node s_b. An ant going from the source node to the destination node, collects information about the quality of the path, and it uses this information to update the pheromone levels of the intermediate nodes, reinforcing the pheromone of the good paths, creating a form of distributed reinforcement learning based on stigmergy [5].

3.1 General Outline of the ACO Based Algorithms

Let us assume that a WSN consists of m static and identical wireless sensors (nodes). The nodes are uniformly distributed in a flat region. The communication area covered by each node is represented by a circle whose radius is r.

A WSN is formally described as a weighted undirected graph *G(V, E, L)*. Where

- V is the set of sensor nodes, $V = \{s_1, s_2, s_3, ..., s_m\}$.
- L is the set of weights.
- E is the set of edges, $E \subset V \times V \times L$, for example, for any $s_i, s_j \in V$, $i \neq j$, $(s_i, s_j, \psi_{ij}(t)) \in E$, where $\psi_{ij}(t))$ is the cost to deliver a data package from s_i to s_j in the time t, in this case $\psi_{ij}(t)$ is the pheromone between s_i and s_j.

Any node s_i has a set of neighbors, defined by:

$$N(s_i) = \{s_j | s_j \in V, d_{ij} \leq r\} \tag{1}$$

r is the wireless communication coverage of the nodes. d_{ij} is the distance between s_i and s_j where the coordinates of the node s_i are x_i and y_i, and the coordinates of the node s_j are x_j and y_j and is calculated by:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2}$$

The algorithm is composed of two phases. In the first phase it starts with a set of ants placed in the source node s_o. When the ant k is at the node s_i at instant t, the next-hop node $s_j \in N(s_i)$ will be selected randomly with a probability $P_{ij}^k(t)$ calculated by:

$$P_{ij}^k(t) = \frac{[\psi_{ij}(t)]^\alpha \times [\xi_{ij}(t)]^\beta}{\sum_{s_l \in N(s_i)} [\psi_{il}(t)]^\alpha \times [\xi_{il}(t)]^\beta} \tag{3}$$

Where ξ_{ij} is the location function defined by:

$$\xi_{ij} = \frac{1}{d_{ij}} \tag{4}$$

$\psi_{ij}(t)$ is the level of pheromone between node s_i and node s_j in the time t. α and β are the adjustable weights of $\psi_{ij}(t)$ and $\xi_{ij}(t)$, respectively.

When ant k reaches the destination node s_b phase two begins. The ant k goes back following the same route, depositing an increment of pheromone on that. This increment of pheromone is defined as follows:

$$\psi_{ij}(t+1) = (1 - p(t)) \times \psi_{ij}(t) + \Delta\psi_{ij} \tag{5}$$

Where $p(t)$ refers to the pheromone evaporating rate in the time t and $\Delta\psi_{ij}$ is the pheromone increment on the route between s_i and s_j in the current round travel.

$$\Delta\psi_{ij} = \sum_{k=1}^{n} \Delta\psi_{ij}^k \tag{6}$$

$\Delta\psi_{ij}^k$ is the pheromone that ant k laid on the route between the node s_i and the node s_j in the current round travel giving by.

$$\Delta\psi_{ij}^k = \begin{Bmatrix} \frac{1}{L^k} \text{ if } k \text{ passed from } s_i \text{ to } s_j \\ 0 \qquad\qquad \text{otherwise} \end{Bmatrix} \tag{7}$$

L^k refers to the length of the route founded by ant k. When ant k has returned to the suorce node s_o the ant k is eliminated.

Below we describe the routing algorithms that we are going to compare.

3.2 Ant Colony Optimization-Based Location-Aware Routing for Wireless Sensor Networks (ACLR)

ACLR [14] tries to find an equilibrium between the lifetime and the delay of the transmissions. In ACLR not all the s_i's neighbors are candidates to be selected to be the next-hop, only the nearer neighbors to the destination node are candidates. This set is defined by:

$$C(s_i) = \{s_j | s_j \in N(s_i), d_{jb} \leq d_{ib}\}. \tag{8}$$

Each node in the WSN has a memory block in which the residual energy, the location information of the node, its neighbors and the destination node are stored. Each ant is a mobile agent that has a contraindication list to memorize the nodes traversed by itself in a round travel. This contraindication list avoids to select the nodes which have been traversed by the ant. The algorithm ACLR is composed of two phases. In the first phase ants walk from the node s_o to the node s_b and the second phase is when ants return from s_b to s_o. The two phases are described below:

First phase: every ant follows the proposed routing scheme. The ant k starts to look for a route from the source node s_o to the destination node s_b. The new transition probability formula that allows ant k to select the next node, is defined as follows:

$$P_{ij}^k(t) = \frac{[\psi_{ij}(t)]^\alpha \times [\xi_{ij}]^\beta \times [\eta_{ij}(t)]^\gamma}{\sum_{s_l \in C(s_i)} [\psi_{il}(t)]^\alpha \times [\xi_{ij}]^\beta \times [\eta_{il}(t)]^\gamma} \tag{9}$$

This transition probability formula is composed not only for location ξ_{ij} and pheromone $\psi_{ij}(t)$ metrics but an energy metric $\eta_{ij}(t)$, which tries to maximize the network lifetime. The location function proposed by ACLR, ξ_{ij} is defined by:

$$\xi_{ij} = \left(\frac{d_{ob}}{d_{oi} + d_{ij} + d_{jb}}\right) \times \left(1 - \frac{d_{ij}}{\sum_{s_l \in C(s_i)} d_{il}}\right) \tag{10}$$

Where d_{ob} is the distance between node s_o and node s_b, d_{oi} is the distance between node s_o and node s_i, d_{ij} is the distance between node s_i and node s_j, d_{jb} is the distance between node s_j and node s_b and d_{il} is the distance between node s_i and node s_l. $\eta_{ij}(t)$ is the energy function proposed by ACLR and is defined as follows:

$$\eta_{ij}(t) = \frac{e_j(t)}{\sum_{s_l \in C(s_i)} e_l(t)} \tag{11}$$

If there is not any next-hop neighbor to select, that is $C(s_i)$ is empty, then ant k returns to the previous-hop node. And s_i is added to the contraindication list of the ant k. When ant k reaches the destination node s_b, a route R^k from the source node to the destination node is found by ant k. L^k is the length of R^k. Second phase: Each ant returns to the source node from the destination node along the route R^k. At the meantime, ant k updates the pheromone on each segment of R^k, following the increment of pheromone defined in formula (5). The ACLR defines $\Delta\psi_{ij}^k$ as follows:

$$\Delta\psi_{ij}^k = \left\{ \begin{matrix} \frac{d_{ob \times Q}}{(d_{oi}+d_{ij}+d_{jb})L^k} & \text{if } s_i \text{ to } s_j \in R^k \\ 0 & \text{otherwise} \end{matrix} \right\} \tag{12}$$

Where Q is a constant, d_{ob}, d_{oi}, d_{ij} and d_{jb} have the same meaning as that of formula (10), respectively. L^k is the length of the route that is found by the ant k in the current round travel.

When the ant k has returned to the suorce node s_o, the ant k is eliminated. When all ants have been eliminated, a new iteration of the algorithm is repeated until it reaches a certain number of iterations.

In the experiments conducted in [14], it should be noted that they use an arbitrary value for energy consumption. They assume that it consumes one unit energy to directly deliver a data package between two nodes. They are not careful on the size of the data packages sent between nodes (ants). This size is vital to calculate the energy consumption in the WSN. Since the data packages they use are of variable size, the energy consumption they calculate is not correct, because it is not the same energy consumption for delivery small data packages that large data packages. To be able to make a fair comparison between algorithms, we must be careful in the way of calculate the energy consumption of each algorithm. Below it is described the following routing algorithm that we will compare.

3.3 Energy-Efficient Ant-Based Routing Algorithm (EEABR)

EEABR [2] is an ACO based routing protocol for WSN, which considers the energy efficiency of the underlying algorithm in order to maximize the networks lifetime. It has been proved that the tasks performed by the sensor nodes that are related with communications (transmitting and receiving data), spend much more energy than those related with data processing and memory management. Since one of the main concerns in WSNs is to maximise the lifetime of the network, it would be preferable that the routing algorithm could perform as much processing as possible in the network nodes, than transmitting all data through the ants to the base station to be processed there. In fact, EEABR tries to minimize the data package size as much as posible.

This algorithm makes a strong difference between forward ants (from the source node s_o to the destination node s_b) and backward ants (from the destination node s_b to the source node s_o). The ant's memory R^k only remember the last two nodes visited, this allows to have a constant ant size. When the ant k pass through the node s, the node s is the responsible of memorize the next information of the ant k: the previous visited node, the forward node, the ant identification and a timer. EEABR is composed by two phases: first phase concerning the forward ants and second phase concerning the backward ants.

First phase (forward ants): when a node s received an ant k, the node s looks into its memory and searches the ant identification in order to avoid the creation of a possible loop. If there is not such record, the node s saves the required information of k and initializes a timer for the ant k. The ant k decides which node is the most convenient to jump to, according to the transition probability formula that is computed as follows:

$$P_{ij}^k(t) = \frac{[\psi_{ij}(t)]^\alpha \times [\eta_{ij}(t)]^\gamma}{\sum_{s_l \in C(s_i)} [\psi_{il}(t)]^\alpha \times [\eta_{il}(t)]^\gamma} \tag{13}$$

Where $\psi_{ij}(t)$ is the level of pheromone between the node s_i and the node s_j, calculated in the same way that (5). α and γ are the adjustable weights of $\psi_{il}(t)$ and $\eta_{il}(t)$, respectively. $\eta_{ij}(t)$ is the energy function defined as follows:

$$\frac{1}{C - e_j} \tag{14}$$

C is the initial energy level of the nodes, and e_j is the energy of sensor j. If the ant k had passed through the node s, then the ant k is eliminated. When the ant k reaches the destination node s_b, a backward ant is created with the forward ant identification as well the forward ant memory.

Second phase (backward ants): the amount of pheromone that the backward ant will lay is calculated according to the EEABR proposed increment of pheromone of the ant k that is defined by:

$$\Delta\psi_{ij}^k = \left\{ \begin{matrix} \frac{1}{\left(C - \left[\frac{Emin^k - Fd^k}{Eavg^k - Fd^k}\right]\right) \times \varphi Bd^k} & k \text{ passed } s_i, s_j \\ 0 & \text{otherwise} \end{matrix} \right\} \tag{15}$$

Where $Emin^k$ is the minimum energy level registered by the forward ant k through its route. $Eavg^k$ is the average energy of the visited nodes so far registered by forward ant k through its route. Fd^k represents the number of nodes that the forward ant k has visited so far. φ is a coefficient and Bd^k is the traveled distance (the number of visited nodes) by the backward ant k until node i.

The timer is used to delete the record that identifies the backward ant, if for any reason the ant does not reach that node within the time defined by the timer. It is important to note that the key point of EEABR algorithm is to minimize the size of data packages transmitted between nodes. Therefore, to make a real and fair comparison between the ACLR and EEABR algorithms, we must be careful in the way of calculate the energy consumption.

4 Experiments

In this Section we present the experimental comparison between the algorithms ACLR and EEABR. To conduct our experiments, we consider the nodes Mica2 Motes of the company Crossbow[1]. We know that the energy required to transmit one bit between two nodes is 4.28 $\mu joules$ and the energy required to receive one bit is 2.36 $\mu joules$ [13] . It is a fact that nodes consume energy simply because of being on, even off or asleep, however, in our simulation the nodes are always on, so this consumption of energy can be disregarded and only take into account the energy consumption to transmit and receive data packages. With these energy consumption values, it can be established that the comparison between the protocols ACLR and EEABR would be more in line with reality and more fair than the one made in their original proposals [14,2]. For this sake, we execute the experiments using three different scenarios that were explained in [2]. We also evaluate three metrics used in [14] to determine the performance of the routing algorithms. These scenarios and metrics are defined as follows:

Scenarios. ***First scenario:*** all nodes start with the same initial energy level, there is only one source node s_o and the destination node s_b is fixed. ***Second scenario:*** all nodes begin with the same initial energy level, the source node s_o changes randomly at each iteration and the destination node s_b is fixed. ***Third scenario:*** the nodes's initial energy level is randomly selected, the source node s_o changes randomly at each iteration and the destination node s_b is fixed.

Metrics. ***Energy Consumption:*** this metric refers to the total used energy in the network in the process of finding the optimal routes from the source node s_o to the destination node s_b. To be fair in the comparision, we use the energy consumption per bit that is shown in [13], for the transmitting and receiving data in each node. As mentioned in the previous section, the energy required to transmit one bit between two nodes is 4.28 $\mu joules$ and the energy required to receive one bit is 2.36 $\mu joules$. ***Latency:*** the time it takes a data package to be sent from the source node s_o to the destination node s_b it is called latency, it is the sum of temporal delays into the network. This metric is commonly calculated as the average of number of nodes visited per route. ***Energy Efficiency:*** it refers to the ratio of the number of data packages received at s_b by the total consumed energy.

Parameter values. The parameters' values are the same for both algorithms, established as follows: The deployed field has a surface area of 300 x 200 (m^2).

[1] "http://www.xbow.com/"

10000 nodes deployed uniformly. Number of ants equal to 20. Wireless communications radius of sensors is r=30 (m). $\alpha = 1, \gamma = 1$. For the algorithm ACLR $\beta = 1$ and $Q = 1$. For the algorithm EEABR $\varphi = 1$ and the pheromone evaporating rate $p(t)$ equal to 0.95. The initial pheromone level for every pair of adjacent nodes is set to $\psi_{ij}(0) = 0.01$.

The energy consumption per bit transmited (as defined by [13]) is 4.28 $\mu joules$, and per bit received is 2.36 $\mu joules$. We performed 50 iterations in each experiment and for both algorithms we ran 20 independent experiments for each scenario.

5 Results

The metric results for the first scenario are shown in Figures 1, 2 and 3 for Energy Consumption, Latency and Energy Efficiency, respectively. It seems that each routing algorithm has an especial focus, clearly the algorithm EEABR outperforms the Energy Consumption in a better way than ACLR, this is a result of its ant sizes, meanwhile the algorithm ACLR outperforms by some (2 or 3 nodes) the EEABR in all the 20 experiments in the Latency metric, wich means that ACLR focuses on minimize the delay in data transmision. In the Energy Efficiency metric, EEABR is more efficient than ACLR.

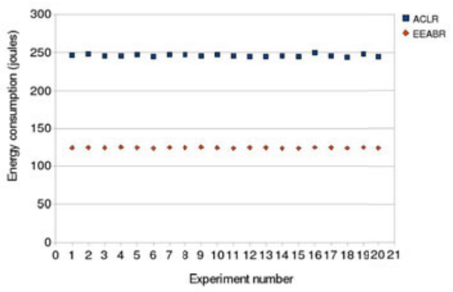


Fig. 1. First scenario. Energy consumption.

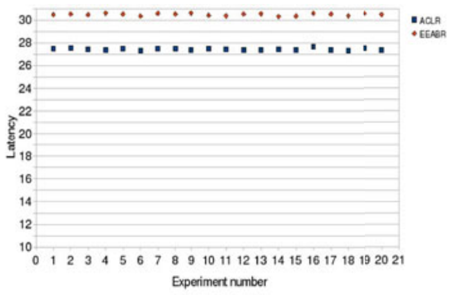


Fig. 2. First scenario. Latency.

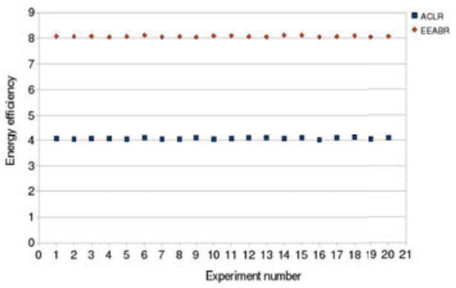


Fig. 3. First scenario. Energy efficiency.

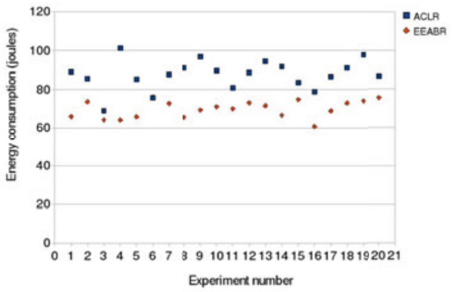


Fig. 4. Second scenario. Energy consumption.

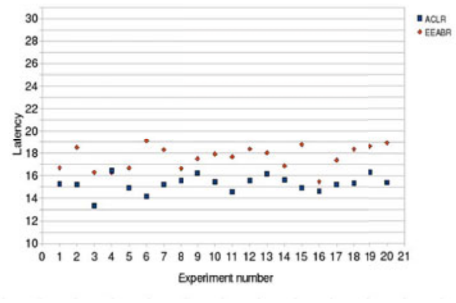


Fig. 5. Second scenario. Latency

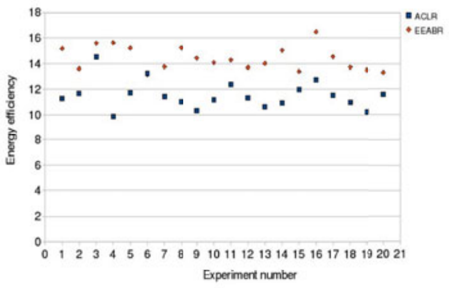


Fig. 6. Second scenario.Energy efficiency

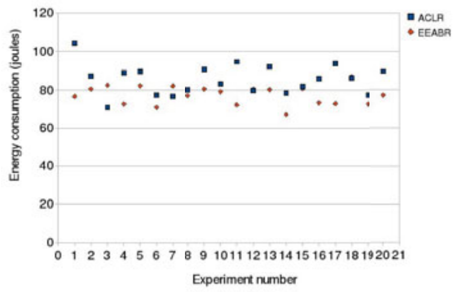


Fig. 7. Third scenario. Energy consumption

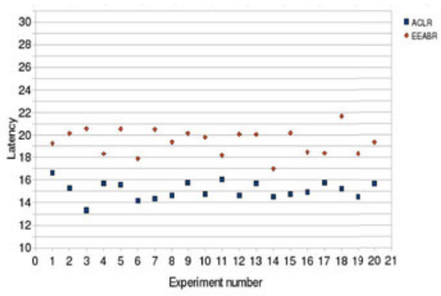


Fig. 8. Third scenario. Latency

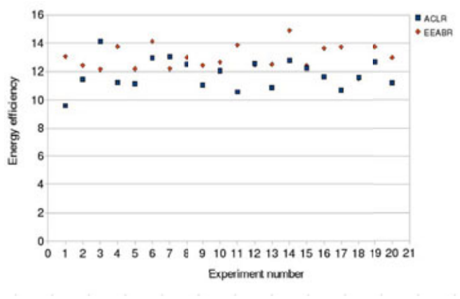


Fig. 9. Third scenario. Energy efficiency

The results for Energy Consumption, Latency and Energy Efficiency metrics are shown in Figures 4, 5 and 6, respectively for the second scenario. For the Energy Consumption metric in this second scenario, the EEABR outperforms ACLR in most of the cases (18 out of 20) with a difference less than the previous scenario. Both algorithms present better values in general than their respective metric in the first scenario. This is because the source node is constantly changing its position, therefore the found paths are more diverse. For the Latency metric case, ACLR outperforms EEABR for most of the experiments. Furthermore, notice that in this second scenario both algorithms perform better than the first scenario. For the Energy Efficiency metric we can observe again that EEABR is better than the ACLR (18 out of 20). Again, the results for this second scenario are consistent with those from the first scenario.

In the third scenario, the metric values for Energy Consumption, Latency and Energy Efficiency metrics are shown in Figures 7, 8 and 9, respectively. We can observe that the Energy Consumption values are a little higher for both algorithms than their respective metric in the second scenario, with similar performance between them. Notice that the latency values for both algorithms are similar in this scenario and similar to the second scenario as well and still ACLR outperforms EEABR. Both algorithms have similar performance for the Energy Efficiency metric in this scenario, being a little bit better EEABR than ACLR (12 out of 20).

6 Conclusion

We can conclude, in general terms, the ACLR and EEABR presented good performances in terms of the defined metrics. However, ACLR provides better results finding shorter routes than EEABR, i.e. two or three nodes less on average. EEABR has shown better performance in Energy Consumption. If the source node is changing its position, both algorithms presented better Energy Consumption than the case when it is fixed. For the scenarios where the source node is changing, the algorithms show better Latency performance than scenarios where the source node is fixed. The algorithms are more efficient, in terms of Energy Efficiency, if the source node is changing for each iteration. For the third scenario, where the nodes's initial energy values are set to different values, the Energy Consumption is very similar for both algorithms, as well the Energy Efficiency. In general, the algorithms present very consistent performance, however EEABR is more efficient maximizing the network's lifetime. This efficiency is due to the usage of slim ants. The results shown that the bio-algorithms to solve the routing problem in Wireless Sensor Networks are viable options.

Acknowledgements

The authors acknowledge support from CONACYT through project 107688 and SIP-IPN 20100273.

References

1. Cai, W., Jin, X., Zhang, Y., Chen, K., Wang, R.: ACO Based QoS Routing Algorithm for Wireless Sensor Networks. In: Ma, J., Jin, H., Yang, L.T., Tsai, J.J.-P. (eds.) UIC 2006. LNCS, vol. 4159, pp. 419–428. Springer, Heidelberg (2006)

2. Camilo, T., Carreto, C., Silva, J.S., Boavida, F.: An Energy-Efficient Ant-Based Routing Algorithm for Wireless Sensor Networks. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 49–59. Springer, Heidelberg (2006)
3. Dorigo, M., DiCaro, G.: Ant Net: A Mobile Agents Approach to Adaptive Routing Technical. IRIDIA Free Brussels University, Belgium (1997)
4. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
5. Farooq, M., Caro, G.A.: Routing Protocols for Next-Generation Networks Inspired by Collective Behaviors of Insect Societies An Overview. Swarm Intelligence, 101–160 (2008)
6. Heimfarth, T., Janacik, P.: Experiments with Biologically-Inspired Methods for Service Assignment in Wireless Sensor Networks. Biologically-Inspired Collaborative Computing 268, 71–84 (2008)
7. Heinzelman, W., Balakrishnan, H.: Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In: Proceedings of the 33rd Hawaii International Conference on System Sciences (2000)
8. Huo, H., Gao, D., Niu, Y., Gao, S.: ASDP An Action-Based Service Discovery Protocol Using Ant Colony Algorithm in Wireless Sensor Networks. In: Zhang, H., Olariu, S., Cao, J., Johnson, D.B. (eds.) MSN 2007. LNCS, vol. 4864, pp. 338–349. Springer, Heidelberg (2007)
9. Iyengar, S., Wu, H., Balakrishnan, N., Chang, S.: Biologically Inspired Cooperative Routing for Wireless Mobile Sensor Networks. IEEE Systems 1(1), 29–37 (2007)
10. Lindsey, S., Raghavendra, C.: PEGASIS: Power-Efficient Gathering in Sensor Information Systems. IEEE Aerospace Conference Proceedings, 1125–1130 (2002)
11. Okdem, S., Karaboga, D.: Routing in Wireless Sensor Networks Using an Ant Colony Optimization ACO Router Chip. Sensors, 909–921 (2009)
12. Reza, G., Rahman, A., Gueaieb, W., Saddik, A.: Ant Colony-Based Reinforcement Learning Algorithm for Routing in Wireless Sensor Networks. IEEE (1-4244-0589-0) 1–6 (2007)
13. Torres, M.G.: Energy Consumption in Wireless Sensor Networks Usig GSP. Master's thesis, Universidad Pontificia Bolivariana, Medellín, Colombia (2006)
14. Wang, X., Li, Q., Xiong, N., Pan, Y.: Ant Colony Optimization-Based Location-Aware Routing for Wireless Sensor Networks. In: Li, Y., Huynh, D.T., Das, S.K., Du, D.-Z. (eds.) WASA 2008. LNCS, vol. 5258, pp. 109–120. Springer, Heidelberg (2008)
15. Wen, Y., Chen, Y., Qian, D.: An Ant-based approach to Power-Efficient Algorithm for Wireless Sensor Networks. 1546–1550 (2007)
16. Ye, N., Shao, J., Wang, R., Wang, Z.: Colony Algorithm for Wireless Sensor Networks Adaptive Data Aggregation Routing Schema. In: Li, K., Fei, M., Irwin, G.W., Ma, S. (eds.) LSMS 2007. LNCS, vol. 4688, pp. 248–257. Springer, Heidelberg (2007)
17. Zhu, X.: Pheromone Based Energy Aware Directed Diffusion Algorithm for Wireless Sensor Network. In: Huang, D.-S., Heutte, L., Loog, M. (eds.) ICIC 2007. LNCS, vol. 4681, pp. 283–291. Springer, Heidelberg (2007)

Approximating Multi-Objective Hyper-Heuristics for Solving 2D Irregular Cutting Stock Problems

Juan Carlos Gomez and Hugo Terashima-Marín

Tecnológico de Monterrey
Monterrey, Nuevo León 64849, México
juancarlos.gomez@invitados.itesm.mx, terashima@itesm.mx

Abstract. This article presents a method based on the multi-objective evolutionary algorithm NSGA-II to approximate hyper-heuristics for solving irregular 2D cutting stock problems under multiple objectives. In this case, additionally to the traditional objective of minimizing the number of sheets used to fit a finite number of irregular pieces, the time required to perform the placement task is also minimized, leading to a bi-objective minimization problem with a tradeoff between the number of sheets and the time required for placing all pieces. We solve this problem using multi-objective hyper-heuristics (MOHHs), whose main idea consists of finding a set of simple heuristics which can be combined to find a general solution for a wide range of problems, where a single heuristic is applied depending on the current condition of the problem, instead of applying a unique single heuristic during the whole placement process. The MOHHs are approximated after going through a learning process by mean of the NSGA-II, which evolves combinations of condition-action rules producing at the end a set of Pareto-optimal MOHHs. We tested the approximated MMOHHs on several sets of benchmark problems, having outstanding results for most of the cases.

Keywords: Hiper-Heuristics; Multi-Objective Optimization; Cutting; Evolutionary Computation.

1 Introduction

Bin packing and cutting stock problems are well-known classical problems with many applications in different areas like operational research, logistics, engineering and related subjects. The basic idea and main goal consists of fitting a finite number of pieces into a minimum number of bins, subject to a practical set of restrictions and requirements. For small combinatorial problems, exact methods like linear programming can be applied. However, when larger and more complex problems appear, exact solutions are not a reasonable choice since the search space grows exponentially, and so does the time to find the optimal solution. Various heuristic and approximate approaches that guarantee finding near optimal solutions have been proposed [11]. However, no reliable method that can solve a large variation of instances of a given problem has been found. In general, methods work well for a few instances, but are deficient for many others.

A hyper-heuristic is a method used to define a high-level heuristic that controls low-level heuristics [1]. The hyper-heuristic decides when and where to apply each single low-level heuristic, depending on the given problem state and the search space. In recent work, based on the research by Ross et al. [13], evolutionary approaches have been used to generate hyper-heuristics for the 2D regular and irregular cutting stock problems [15][16]. These methods assemble a combination of single heuristics (each concerned with selection and placement of a piece), and this combination takes into account the quality of partial solutions provided by the single heuristics.

Nevertheless, the majority of the works devoted to the problems of bin packing and cutting stocks are focused on mono-objective solutions like minimizing the trim loss; however these problems are naturally multi-objective, since several (opposite) objectives can be optimized at the same time. Just recently some works devoted to multi-objective cutting and packing problems have started to emerge like [7][12]; the present article intends to contribute with another perspective of solution for these problems. In this work we are focused on problems involving cutting 2D irregular pieces where two objectives need to be minimized: the number of sheets used to cut a finite number of pieces with irregular shapes and the time required to perform the placement of all the pieces inside the sheet.

The aim of this paper is to present a method based on the Multi-Objective Evolutionary Algorithm (MOEA) [3] NSGA-II [4] to approximate generalized Multi-Objective Hyper-Heuristics (MOHH) in order to solve the cutting-stock problem describe above. We use NSGA-II with a variable-length representation. This algorithm evolves combinations of condition-action rules through a learning process, producing at the end a set of Pareto-optimal MOHHs. Finally, we test the approximated MOHHs on several sets of benchmark problems. Results of the proposed model in the 2D cutting-stock problem are truly encouraging.

The remainder of this paper is organized as follows. Section 2 describes the multi-objective cutting-stock problem. Section 3 presents the proposed solution based on MOHH. This is followed by the experimental setup, the results and discussion in section 4. Finally, in section 5 we include our conclusions and some ideas for future work.

2 Description of the Problem

The Cutting-Stock and Packing Problems (CuSPPs) are among the earliest problems in the literature of operational research. In 1939, Kantorovich [10] studied applications in all the industries whose products were in a flat sheet form. An extensive literature on the CuSPPs and their applications has developed since. For example, Golden in [8] gives an abstract description of a variety of different solution methods; in [2] Chen et al. discuss a number of solution methods, and Dyckhoff in [5] lists a number of solution methods and applications and presents a systematic categorization of cutting and packing problems.

Here we solve instances of problems considered as Single Bin-Size 2D Cutting-Stock Problems with Irregular Shape Pieces. We can define formally these problems as: given a set $L = (a_1, a_2, ...a_n)$ of pieces to be cut, each one of size $s(a_i) \in (0, A_o]$, from a set m of cutting stock sheets of size A_o, the multi-objective goal of cutting the pieces from the sheets can then be modeled as follows.

$$\text{minimize } z_1 = \sum_{i=1}^{m} y_i \tag{1}$$

$$\text{minimize } z_2 = \sum_{i=1}^{n} t_i \tag{2}$$

$$\text{s.t. } \textstyle\sum_{i=1}^{n} s(a_i) \leq mA_o$$

where expression 1 minimizes the number y_i of needed sheets, and expression 2 minimizes the time to place all the pieces inside the sheets, where t_i indicates the required time to put the piece a_i inside a sheet, using a given heuristic.

Intuitively, the two objective functions are of conflicting nature. While a large number of sheets allows the placement of all the pieces very fast, having $z_2 \rightarrow 0$, the trim of material will be also large; on the other hand, a solution with $z_1 \rightarrow 0$, will tend to require more time to place all the pieces in the best position in order to do not waste material. This means that not a single solution x exists in the set of feasible solutions X that equally minimizes both objective functions z_1 and z_2. Then, we have a vector optimization problem in which a solution $x \in X$ is evaluated with respect to a vector $Z(x) = (z_1(x), z_2(x))$. The solution of the problem has consequently to be seen in the identification of all efficient outcomes or the Pareto-set P, defined as follows:

Definition 1. ***(Dominance):*** *$Z(x)$ is said to dominate $Z(x')$ iff $z_k(x) \leq z_k(x')$ $\forall k = 1, \ldots, K \wedge \exists k | z_k(x) < z_k(x')$. We denote the dominance of $Z(x)$ over $Z(x')$ with $Z(x) \preceq Z(x')$.*

Definition 2. ***(Efficiency, Pareto-optimality):*** *The vector $Z(x), x \in X$ is said to be efficient iff $\neg\exists Z(x'), x' \in X | Z(x') \preceq Z(x)$. The corresponding alternative x is called Pareto-optimal and the set of all Pareto-optimal alternatives is the Pareto-set P.*

3 Multi-Objective Hyper-Heuristics Solution Approach

Hyper-heuristics deal with the process of choosing good single heuristics for solving the problem at hand. The idea is to discover a combination of single heuristics that can perform well on a whole range of problems and in such a way that one heuristic's strengths make up for the drawbacks of another [14]. The rationale is that there is no a unique best single heuristic to solve well all the instances of a problem, since certain problems may contain features that would enable a specific heuristic to work well, but those features may not be present in other problems. Then, a combination of heuristics, selectively applied based on the features present in a problem, may work well on a wide range of problems.

3.1 Single Heuristics

For the 2D irregular cutting-stock problem, the related single heuristics must define the exact location of the pieces inside the sheet. In this work two kinds of single heuristics to be combined by the Multi-Objective Hyper-Heuristics (MO-HHs) were considered: one kind for selecting the pieces and sheets, and the other for placing the pieces into the sheets. The selection heuristics are shown below. Some of these heuristics are described in more detail in [13] and [9].

- **First Fit (FF).** Consider the opened sheets in turn in a fixed order and place the piece in the first one it fits.
- **First Fit Decreasing(FFD).** Sort pieces in decreasing order, and the largest one is placed according to FF.
- **First Fit Increasing (FFI).** Sort pieces in increasing order, and the smallest one is placed according to FF.
- **Filler + FFD.** Places as many pieces as possible within the open sheets. If at least one piece has been placed, the algorithm stops. The FFD algorithm is applied, otherwise.
- **Next Fit (NF).** Use the current sheet to place the next piece, otherwise open a new one and place the piece there.
- **Next Fit Decreasing (NFD).** Sort the pieces in decreasing order, and the largest one is placed according to NF.
- **Best Fit (BF).** This places the piece in the opened sheet where it best fits (i.e. with the minimum waste).
- **Best Fit Decreasing (BFD).** Same as the previous one, but sorting the pieces in decreasing order.
- **Worst Fit (WF).** It places the piece in the opened sheet where it worst fits (i.e. with the largest waste).
- **Djang and Fitch (DJD).** It places pieces in a sheet, taking pieces by decreasing size until the sheet is at least one-third full. Then, it initializes w, a variable indicating the allowed waste, and looks for combinations of 1,. . .,5 pieces producing a waste w. If any combination fails, it increases w accordingly.

The placement heuristics, described in detail in [17], are the following:

- **Bottom-Left (BLI).** The piece starts at the top right corner of the sheet and it slides down and left with a sequence of movements until no other movement is possible. If the final position does not overlap the sheet boundaries, the piece is placed in that position. The heuristic does not allow a piece to skip around another placed piece. It's a simple and fast heuristic.
- **Constructive Approach (CA).** The heuristic starts by placing the first piece at the bottom and left of the sheet. Then, the next piece is placed in one of the five positions: $(\overline{x}, 0), (0, \overline{y}), (\underline{x}, \overline{y}), (\overline{x}, \overline{y})$and $(\overline{x}, \underline{y})$, where $\overline{x}, \underline{x}, \overline{y}$, and $\underline{y}$ are the maximum and minimum coordinates in x and y in relation to the first piece. For each position, the next piece slides following down and left movements, and the one that places the piece deepest (bottom and left) is chosen, except in special cases such as when a hole is formed.

- **Constructive-Approach (Minimum Area) (CAA).** In this modification of the previous heuristic, the best position from the list is selected based on which one yields the bounding rectangle with minimum area, containing all pieces, and that fits in the bottom left corner of the object.
- **Constructive-Approach (Maximum Adjacency) (CAD).** With the first piece only the four corners of the sheet are considered. For the subsequent pieces, the possible points are the same than in CA. For each position in the list the piece starts in that position and its adjacency (i.e. the common boundary between its perimeter and the placed pieces and the sheet edges) is computed. Then, the piece is slid down and left and the adjacency is computed again. The position with the largest adjacency is selected as the position of the new piece.

3.2 NSGA-II

In this paper we use the NSGA-II [4], a MOEA [3] with variable length chromosomes as the approach to approximate generalized MOHHs to solve the cutting-stock problem as described above.

The NSGA-II (Elitist Non-Dominated Sorting Genetic Algorithm) is a well known and with good behavior MOEA proposed by Deb et al. in [4], used as a reference for many works in multi-objective optimization. In this algorithm, initially a random parent population P_0 of size N is created. The population is sorted based on the non-domination. Each solution is assigned a fitness (or rank) equal to its non-domination level (1 is the best level, 2 is the next-best level, and so on). Thus, minimization of this rank is assumed. The normal binary tournament selection, recombination, and mutation operators are used to create an offspring population Q_0 of size N. After that, a combined population $R_t = P_t + Q_t$ is formed. Next, the population R_t is sorted according to non-domination, forming the fronts $F = F_1, F_2, \ldots$. Since all previous and current population members are included in R_t, elitism is ensured. Now, solutions belonging to the best non-dominated set F_1 are of best solutions in the combined population and must be emphasized more than any other solution inside it. If the size of F_1 is smaller than N, all the members of the set F_1 are chosen for the new population P_{t+1}. The remaining members of the population P_{t+1} are chosen from subsequent non-dominated fronts in the order of their ranking. Thus, solutions from the set F_2 are chosen next, followed by solutions from the set F_3, and so on. This procedure is continued until no more sets can be accommodated. The new population P_{t+1} is now used for selection, crossover, and mutation to create a new population Q_{t+1}, and the process is repeated until a stop criteria is satisfied.

3.3 Model

In this work we adapted the evolutionary model proposed in [17], which produces general hyper-heuristics using a simplified representation of the state of the problem. A chromosome in the NSGA-II algorithm inside our model consists of a number of points in this simplified state space, each point being labeled with a given heuristic. Then, if we have a problem state S in the simplified state space, we find the nearest point in the chromosome and apply the heuristic recorded on the points label. This will transform the problem to a new state S', and the process is repeated until a complete solution has been constructed.

A chromosome therefore represents a complete recipe for solving a problem, using this simple algorithm: until the problem is solved, (a) determine the current problem state S, (b) find the nearest point to it, (c) apply the heuristic attached to the point, and (d) update the state. The NSGA-IIs task is to find a set P of Pareto-optimal chromosomes that are capable of obtaining good solutions for a wider variety of problems, taking in consideration the trade off between the two minimization objectives defined in equations 1 and 2; the set of chromosomes P are the multi-objective hyper-heuristics we are seeking.

Representation. Each chromosome is composed by a variable number of blocks. The initial number of blocks is randomly created and after it, the evolutionary process can create or delete blocks. Each block includes nine numbers. The first eight lie in the range 0 to 1 and represent the problem state, so a block can be seen as the labeled point. The label is the ninth number, which identifies a particular action (single heuristic). The NSGA-II's task is to create and evolve a certain number of such labeled points, using the problem-solving algorithm defined above, where to determine the *nearest* point we use Euclidean distance.

In the problem state, the first three numbers are related to rectangularity, a quantity that represents the proportion between the area of a piece and the area of a horizontal rectangle containing it. These numbers represent the fraction of remaining pieces with high rectangularity [0.9,1], medium rectangularity [0.5,0.9] and low rectangularity [0,0.5]. The fourth to seventh numbers are related to the area of pieces, and are categorized as follows (A_o is the sheet area, A_p is the piece area): huge ($A_o/2 < A_p$); large ($A_o/3 < A_p \leq A_o/2$); medium ($A_o/4 < A_p \leq A_o/3$); and small ($A_p \leq A_o/4$). The eighth number represents the fraction of the total number of pieces that remain to be placed. The label is selected from all possible combinations of selection and placement heuristics. Figure 1 shows a graphical representation of a chromosome using the simplified state space. This simplified feature space intends to represent the essential information about the geometry (rectangularity), and occupied and free space of the problem's whole configuration in a given step of the solution process. More features can

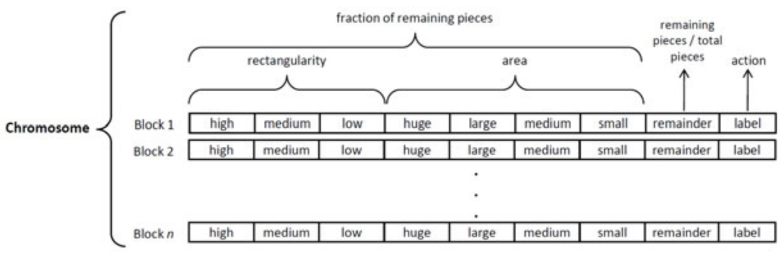


Fig. 1. Graphical representation of a chromosome using the simplified state space

be added or considered but with the corresponding increase in complexity. The current problem's state is compute in each step of the solution process, taking the information stored in the model about the pieces assigned, pieces free and open (available) sheets. Finally, there are 40 different combinations for the action as shown in Table 1.

Table 1. List of possible actions

Action	Selection	Placement	Action	Selection	Placement
1	FF	BLI - Bottom Left	21	NFD	BLI - Bottom Left
2		CA - Constructive	22		CA - Constructive
3		CAA - Constructive-M. Area	23		CAA - Constructive-M. Area
4		CAD - Constructive-M. Adjacency	24		CAD - Constructive-M. Adjacency
5	FFD	BLI - Bottom Left	25	BF	BLI - Bottom Left
6		CA - Constructive	26		CA - Constructive
7		CAA - Constructive-M. Area	27		CAA - Constructive-M. Area
8		CAD - Constructive-M. Adjacency	28		CAD - Constructive-M. Adjacency
9	FFI	BLI - Bottom Left	29	BFD	BLI - Bottom Left
10		CA - Constructive	30		CA - Constructive
11		CAA - Constructive-M. Area	31		CAA - Constructive-M. Area
12		CAD - Constructive-M. Adjacency	32		CAD - Constructive-M. Adjacency
13	Filler+FFD	BLI - Bottom Left	33	WF	BLI - Bottom Left
14		CA - Constructive	34		CA - Constructive
15		CAA - Constructive-M. Area	35		CAA - Constructive-M. Area
16		CAD - Constructive-M. Adjacency	36		CAD - Constructive-M. Adjacency
17	NF	BLI - Bottom Left	37	DJD	BLI - Bottom Left
18		CA - Constructive	38		CA - Constructive
19		CAA - Constructive-M. Area	39		CAA - Constructive-M. Area
20		CAD - Constructive-M. Adjacency	40		CAD - Constructive-M. Adjacency

The Fitness Functions. Each MOHH is evaluated using two fitness functions, one for the waste of space in the sheet and another for the total time needed to place the pieces on the sheet. The waste of space in a given sheet and the corresponding fitness function are defined as:

$$W = 1 - \frac{\sum_{i=1}^{k} s(a_i)}{A_o} \tag{3}$$

$$FF_1 = \frac{\sum_{i=1}^{m} W_i^2}{m} \tag{4}$$

where k is the number of pieces inside the sheet, $s(a_i)$ the size of each piece, A_o the size of the sheet and m is the number of sheets used. The second fitness function is defined as:

$$FF_2 = \sum_{i=1}^{n} t(h(a_i)) \tag{5}$$

where $t(h(a_i))$ is the required time to place the piece a_i using the heuristic h.

During the NSGA-II process, when a new individual (MOHH) is created (in the first parent population or in the subsequent children populations), a set of 5 problems is assigned to it. These problems are selected randomly from the training set. Then, the individual is evaluated with each problem using the previous fitness functions, the fitness values are added and averaged to obtain two final fitness values:

$$FFT_j(MOHH) = \frac{\sum_{i=1}^{5} FF_j(p_i)}{5}; j = 1, 2 \tag{6}$$

where p_i is the i-th problem assigned to the individual and index j indicates the number of fitness function. During the evolution, if an individual from the parent population survives for the next generation, a new problem is assigned to it and the fitness functions are recomputed.

$$FFT_j^l(MOHH) = \frac{(FFT_j^{l-1} * n_p) + FF_j(p)}{n_p + 1}; j = 1, 2 \tag{7}$$

where FFT_j^l is the value of the j-th total fitness function for the generation l, n_p is the number of problems the individual has solved until now and $FF_j(p)$ is the value of the j-th fitness function for the new problem.

This task of assign new problems to old individuals and recompute their fitness values continue during the whole evolutionary process.

Genetic Operators. In this work we use two uniform versions of cross-over and mutation, the first version works at the block level and the second one works with the internal elements of each block. In the block level cross-over a random number of complete blocks is exchanged between two parent individuals to create two children individuals; since the number of blocks in each chromosome (individual) is variable, the random number is less or equal to the shortest chromosome. In the block level mutation, given certain probability, a randomly selected block can be deleted from the chromosome or a randomly created block can be added to the chromosome. In the internal level cross-over and mutation, they select a random number of blocks and for each block a random number of values to be interchanged or mutate, depending on the operator.

4 Experimental Results

4.1 Problem Instances

For this work we generated 18 different types of problems, with 30 instances for each type, totalling 540 instances. Their characteristics can be seen in Table 2. We also added a problem from the literature [6], which was scaled by a factor of 10 in order to have the sheet size of 300 × 300. Instances of type G are the only problems with unknown optimal number of objects, since they were produced after alterations to randomly generated problems with known optimum.

Table 2. Description of problem instances

Type	Sheets	Pieces (size)	Number of Instances	Optimum	Type	Sheets	Pieces (size)	Number of Instances	Optimum
Fu	300 × 300	12	1	unknown	Type J	1000 × 1000	60	30	4
Type A	1000 × 1000	30	30	3	Type K	1000 × 1000	54	30	6
Type B	1000 × 1000	30	30	10	Type L	1000 × 1000	30	30	3
Type C	1000 × 1000	36	30	6	Type M	1000 × 1000	40	30	5
Type D	1000 × 1000	60	30	3	Type N	1000 × 1000	60	30	2
Type E	1000 × 1000	60	30	3	Type O	1000 × 1000	28	30	7
Type F	1000 × 1000	30	30	2	Type P	1000 × 1000	56	30	8
Type G	1000 × 1000	36	30	unknown	Type Q	1000 × 1000	60	30	15
Type H	1000 × 1000	36	30	12	Type R	1000 × 1000	54	30	9
Type I	1000 × 1000	60	30	3					

4.2 Experiments with the Proposed Model

The problem instances are divided into a training and a testing set. The NSGA-II is performed with the training set only, until a termination criterion is met and the set P of general MOHH's has been evolved. All instances in the testing set are then solved with each member of the set P and the results are recorded.

In order to test the overall performance of the model, various experiments were designed, which are described as follows:

- **Experiment Type A.** Instances are divided into two groups: Group A and Group B. Group A is the training set and is formed by instance types A, B, C, D, E, F, G, H and I plus the Fu instance. After running our model over the training set, a set of Pareto-optimal MOHH were obtained and each one was tested with Set B (instance types J, K, L, M, N, O, P, Q and R).
- **Experiment Type B.** This experiment is similar to Experiment Type A, except that the training and testing sets are interchanged.
- **Experiment Type C.** This experiment takes the Fu instance and 15 instances from each problem type (from A through R) to form the training set. The remaining instances form the testing set.
- **Experiment Type IV.** It is the same as Experiment Type C, except that the training and testing sets are swapped

The settings for NSGA-II were: 36 individuals, 80 generations, mutation and cross-over probabilities of 0.1 and 0.9. With this, on average a single run of the algorithm for one of the experiments takes about 12 hours to approximate the Pareto-optimal set P, using a 2.8Ghz Core2Duro PC with 4Gb in RAM.

Table 3. Number of extra objects required for each MOHH for the testing set and the average time for placing all the pieces. Experiment A.

	HH$_{a1}$	**HH$_{a2}$**	**HH$_{a3}$**	**HH$_{a4}$**	**HH$_{a5}$**	**HH$_{a6}$**	**HH$_{a7}$**	**HH$_{a8}$**	**HH$_{a9}$**	**HH$_{a10}$**	**HH$_{a11}$**	**HH$_{a12}$**
Time	2926	312	2	1.8	9.7	33.5	1.6	1817.4	2920	4.8	3.6	3.2
Sheets												
-1	0.4								0.4			
0	90.4	69.6	0.0	0.0	0.0	0.0	0.0	49.3	90.4	0.00	0.0	0.0
1	7.4	19.4	0.4	0.4	0.4	22.2	0.4	42.2	7.4	0.4	13	16.3
2	1.5	0.0	14.4	14.4	14.4	33.7	14.4	8.5	1.1	14.4	32.2	13.7
3	0.0	0.0	7.4	8.2	7.4	21.1	7.4	0.0	0.4	7.4	25.9	9.3
4	0.0	0.0	4	4.4	4.1	10.4	4.1	0.0	0.0	4.1	14.1	13.3
>4	0.4	11.1	73.7	72.6	73.7	12.6	73.7	0.0	0.4	73.7	14.8	47.4

Table 4. Number of extra objects required for each MOHH for the testing set and the average time for placing all the pieces. Experiment B.

	HH$_{b1}$	**HH$_{b2}$**	**HH$_{b3}$**	**HH$_{b4}$**	**HH$_{b5}$**	**HH$_{b6}$**	**HH$_{b7}$**	**HH$_{b8}$**	**HH$_{b9}$**	**HH$_{b10}$**	**HH$_{b11}$**	**HH$_{b12}$**
Time	37.9	1074.9	6.1	3203.8	37.6	258.2	285.2	1.7	2.8	627.5	375.6	224.1
Sheets												
0	9.6	97.4	11.1	97.4	9.2	62.4	69.4	9.2	9.2	80.1	70.5	55.7
1	26.9	2.6	1.1	2.6	27.3	24.4	20.3	10.7	10.7	13.7	19.6	1.5
2	28.8	0.0	12.9	0.0	28.8	8.9	7.0	23.2	23.2	4.8	8.5	3.0
3	19.6	0.0	6.6	0.0	19.9	4.1	3.3	16.2	16.2	1.5	1.5	3.0
4	10.7	0.0	56.1	0.0	10.7	0.4	0.0	10.3	10.3	0.0	0.0	2.6
>4	4.4	0.0	0.0	0.0	4.1	0.0	0.0	30.3	30.3	0.0	0.0	34.3

Tables 3, 4, 5 and 6 show respectively the results for each one of the previously described experiments, where these results are the output of one single run of the complete model for each experiment. The second row of the tables represents the average normalized time a MOHH needs to solve a problem, using its combination of single heuristics. The time is not in real time, but was measured by assigning a time score to each single heuristic, depending on the number of trials or movements ir requires to place a piece inside a sheet; which in fact depends on the current configuration of the previous placed pieces in the sheet. These scores were assigned before the experiments, by solving all the problems with every single heuristic to know the performance of each one for every problem, where the most expensive heuristic has a value of 100, and the rest of the heuristics have a fraction of this time. When applying the MOHH to solve a group of problems, for every heuristic used during the solution process its score is added and averaged at the end.

Taking as baseline the best single heuristic to solve each particular problem, the rest of the rows in the tables show the number of extra sheets a MOHH needs to place all the pieces with respect to that best heuristic. The number in each cell is the percentage of problems where the MOHH needs from -1 to more than 4 sheets to place all pieces with respect to the best single heuristic. It is possible to observe how when a small number of extra sheets (-1 or 0) is needed, the time to perform the placement is big, and the opposite when the placement time is small the number of extra sheets is big, due to the conflicting objectives. This allows for an user to select the best solution in accordance to his particular needs: fastness, precision or an equilibrium.

Table 5. Number of extra objects required for each MOHH for the testing set and the average time for placing all the pieces. Experiment C.

	$\mathbf{HH}_{c1}$	$\mathbf{HH}_{c2}$	$\mathbf{HH}_{c3}$	$\mathbf{HH}_{c4}$	$\mathbf{HH}_{c5}$	$\mathbf{HH}_{c6}$	$\mathbf{HH}_{c7}$	$\mathbf{HH}_{c8}$	$\mathbf{HH}_{c9}$	$\mathbf{HH}_{c10}$	$\mathbf{HH}_{c11}$	$\mathbf{HH}_{c12}$
Time	4.2	276.4	2380.3	297.0	9.8	1.9	279.1	11.6	336.9	2306.7	2.4	234.0
Sheets												
-1			0.4							0.7		1.5
0	4.4	69.6	91.9	69.3	4.8	0.0	66.3	4.8	72.6	90.7	0.0	55.9
1	24.1	24.4	7.8	24.8	24.1	5.9	23.7	18.9	21.5	8.5	6.7	26.7
2	25.9	4.4	0.0	4.8	25.9	13.7	5.6	23.0	5.2	0.0	16.3	7.8
3	24.4	1.1	0.0	1.1	24.8	13.3	4.4	28.1	0.7	0.0	17.4	3.3
4	11.9	0.0	0.0	0.0	11.5	8.5	0.0	14.1	0.0	0.0	17.8	3.3
>4	9.3	0.4	0.0	0.0	8.9	58.5	0.0	11.1	0.0	0.0	41.9	1.5

Table 6. Number of extra objects required for each MOHH for the testing set and the average time for placing all the pieces. Experiment D.

	$\mathbf{HH}_{d1}$	$\mathbf{HH}_{d2}$	$\mathbf{HH}_{d3}$	$\mathbf{HH}_{d4}$	$\mathbf{HH}_{d5}$	$\mathbf{HH}_{d6}$	$\mathbf{HH}_{d7}$	$\mathbf{HH}_{d8}$	$\mathbf{HH}_{d9}$	$\mathbf{HH}_{d10}$	$\mathbf{HH}_{d11}$	$\mathbf{HH}_{d12}$
Time	2.4	70.0	341.0	1.8	1.6	3.9	70.0	333.8	3.9	1.5	341.1	316.1
Sheets												
0	0.4	10.3	72.7	0.4	0.4	4.1	10.3	72.7	4.1	0.0	72.7	70.1
1	8.9	22.1	22.9	5.5	5.5	24.4	22.1	21.8	24.4	5.9	22.9	20.7
2	16.2	31.7	3.7	14.0	13.3	29.9	31.7	3.0	29.9	12.5	3.7	5.2
3	12.5	20.3	0.7	10.0	10.7	20.7	20.3	1.1	20.7	11.4	0.7	2.2
4	15.9	12.5	0.0	7.0	5.5	12.9	12.5	0.7	12.9	5.5	0.0	1.5
>4	46.1	3.0	0.0	63.1	64.6	8.1	3.0	0.7	8.1	64.6	0.0	0.4

Since the NSGA-II tries to find the majority of the elements from the Pareto-optimal set P, the number of MOHHs found by the algorithm is sometimes big. Because a lack of space, we present a subset of 12 MOHHs from P, by splitting the set in 4 fronts and take 3 solutions from each one; trying in this way to preserve the diversity. Except for the experiment D, where the model found solutions with more balance between the two objectives, the rest of the experiments produce a broad variation in solutions, with times scores ranging from 1 to 3000, with the corresponding increase or decrease of needed sheets.

Given the relative novelty of hyper-heuristics and multi-objective cutting and packing area, our results are in an early stage to be compared with other techniques, because there are still few works in the area, and no one about irregular cutting using the same set of problems to make feasible a direct comparison. For that reason we present the results of MOHHs in comparison with the baseline of the single heuristics.

5 Conclusions and Future Work

In this paper we have described experimental results of a model based on the MOEA NSGA-II which evolves combinations of condition-action rules representing problem states and associated selection and placement heuristics for solving multi-objective 2D irregular cutting-stock problems, where we wanted the (opposite) objectives of minimizing the number of sheets used to cut the set of pieces and the total time to place the pieces inside the sheets. These combinations found by the model are called Multi-Objective Hyper-Heuristics (MOHHs). In general, the model efficiently approximate the set of Pareto-optimal MOHHs after going through a training phase, and when applied to an unseen testing set, those MOHHs solve the problems very efficiently taking in to account the trade-off between the two objectives. Ideas for future work involve an analysis of frequencies of use for each single heuristic inside each approximated MOHH, to better understand which single heuristics are more employed during the solution process, and probably approximate a more general MOHH from there; extending the proposed strategy to solve problems with more complex structure like 3D packing problems; including other objectives to be minimized, like the balance weight, the number of cuts or the heterogeneousness inside a sheet; or including other kinds of pieces with arbitrary shapes.

Acknowledgment

This research was supported in part by ITESM under the Research Chair CAT-144 and the CONACYT Project under grant 99695 and the CONACYT post-doctoral grant 290554/37720.

References

1. Burke, E., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S.: Hyper-heuristics: An emerging direction in modern research technolology. In: Handbook of Metaheuristics, pp. 457–474. Kluwer Academic Publishers, Dordrecht (2003)

2. Cheng, C.H., Fiering, B.R., Chang, T.C.: The cutting stock problem. A survey. International Journal of Production Economics 36(3), 291–305 (1994)
3. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, Dordrecht (2002)
4. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A fast elitist nondominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Proceedings of Parallel Problem Solving From Nature VI Conference, pp. 849–858 (2000)
5. Dyckhoff, H.: A topology of cutting and packing problems. European Journal of Operational Research 44(2), 145–159 (1990)
6. Fujita, K., Akagji, S., Kirokawa, N.: Hybrid approach for optimal nesting using a genetic algorithm and a local minimisation algorithm. In: Proceedings of the Annual ASME Design Automation Conference 1993, Part 1 (of 2), pp. 477–484 (1993)
7. Geiger, M.J.: Bin packing under multiple objectives - a heuristic approximation approach. In: International Conference on Evolutionary Multi-Criterion Optimization 2008, pp. 53–56 (2008)
8. Golden, B.L.: Approaches to the cutting stock problem. AIIE Transactions 8(2), 256–274 (1976)
9. Hopper, E., Turton, B.C.: An empirical study of meta-heuristics applied to 2D rectangular bin packing. Studia Informatica Universalis 2(1), 77–106 (2001)
10. Kantorovich, L.V.: Mathematical methods of organizing and planning production. Management Science 6(4), 366–422 (1960)
11. Lodi, A., Martella, S., Monaci, M.: Two-dimensional packing problems: A survey. European Journal of Operational Research 141(2), 241–252 (2002)
12. Muñoz, C., Sierra, M., Puente, J., Vela, C.R., Varela, R.: Improving cutting-stock plans with multi-objective genetic algorithms. In: International Work-conference on the Interplay between Natural and Artificial Computation 2007, pp. 528–537 (2007)
13. Ross, P., Schulenburg, S., Blázquez, J.M., Hart, E.: Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. In: Proceedings of the Genetic and Evolutionary Computation Conference 2002, pp. 942–948 (2002)
14. Ross, P.: Hyper-heuristics. In: Search Methodologies: Introductory Tutorials in Optimization and Decision Support Methodologies, ch.17, pp. 529–556. Springer, Heidelberg (2005)
15. Terashima-Marín, H., Flores-Álvarez, E.J., Ross, P.: Hyper-heuristics and classifier systems for solving 2D-regular cutting stock problems. In: Proceedings of the Genetic and Evolutionary Computation Conference 2005, pp. 637–643 (2005)
16. Terashima-Marín, H., Farías-Zárate, C.J., Ross, P., Valenzuela-Rendón, M.: A GA-based method to produce generalized hyper-heuristics for the 2D-regular cutting stock problem. In: Proceedings of the Genetic and Evolutionary Computation Conference 2006, pp. 591–598 (2006)
17. Terashima-Marín, H., Ross, P., Farías-Zárate, C.J., López-Camacho, E., Valenzuela-Rendón, M.: Generalized hyper-heuristics for solving 2D regular and irregular packing problems. Annals of Operations Research (2008)

Particle Swarm Optimization with Gravitational Interactions for Multimodal and Unimodal Problems

Juan J. Flores[1], Rodrigo López[1], and Julio Barrera[2]

[1] Universidad Michoacana de San Nicolás de Hidalgo
División de Estudios de Posgrado, Facultad de Ingeniería Eléctrica
[2] CINVESTAV-IPN
Departamento de Computación
Evolutionary Computation Group
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D.F. 07360, Mexico
juanf@umich.mx, rlopez@faraday.fie.umich.mx, julio.barrera@gmail.com

Abstract. Evolutionary computation is inspired by nature in order to formulate metaheuristics capable to optimize several kinds of problems. A family of algorithms has emerged based on this idea; e.g. genetic algorithms, evolutionary strategies, particle swarm optimization (PSO), ant colony optimization (ACO), etc. In this paper we show a population-based metaheuristic inspired on the gravitational forces produced by the interaction of the masses of a set of bodies. We explored the physics knowledge in order to find useful analogies to design an optimization metaheuristic. The proposed algorithm is capable to find the optima of unimodal and multimodal functions commonly used to benchmark evolutionary algorithms. We show that the proposed algorithm works and outperforms PSO with niches in both cases. Our algorithm does not depend on a radius parameter and does not need to use niches to solve multimodal problems. We compare with other metaheuristics respect to the mean number of evaluations needed to find the optima.

Keywords: Optimization, gravitational interactions, evolutionary computation, metaheuristic.

1 Introduction

Multimodal optimization problems deal with objective functions that commonly contain more than one global optima and several local optima. In order to find all the global optima in multimodal problems with classical methods, one typically runs a given method several times with different starting points, expecting to find all the global optima. However, these techniques do not guarantee the location of all optima. Therefore, this kind of techniques are not the best way to explore multimodal functions with complex and large search spaces. In the evolutionary computation literature exists a variety of metaheuristics challenging the typical problems of classical optimization. E.g. In particle swarm optimization with niches; the best particle makes a niche with all particles within a radius r, until the niche is full; it then selects the next best no niched and its closets particles to form the second niche; the process until all particles are assigned to a niche. Objective function stretching, introduced by Parsopolous [1], [7] is another algorithm whose strategy is to modify the fitness landscape in order to remove local optima and avoid the premature convergence in PSO. In a minimization problem, a possible local minimum is stretched to overcome a local maximum allowing to explore other sections of the search space identifying new solutions. GSA introduced by Rashedi [5], is a gravitational memory-less (does not include a cognitive component in the model) metaheuristic capable to find only one global optima in unimodal and multimodal problems with more than one global optima, where a heavier mass means a better solution and the gravitational constant G is used to adjust the accuracy search.

In our work we explore the properties of gravitational interactions in order to make an useful metaheuristic to find optima in unimodal and multimodal problems. In Section 2 addresses the main motivation of our work: The Newton's Law of Universal Gravitation. In Section 3 we define the Gravitational Interactions Optimization (GIO) metaheuristic for unimodal and multimodal functions. Section 4 presents to the GIO metaheuristic with differents unimodal and multimodal problems. Section 5 presents the conclusions of this work.

2 Newton's Law of Universal Gravitation

The attraction force of two particles is proportional to their masses and inversely proportional to their distance. The Law of Universal Gravitation was proposed by Isaac Newton [10]. This law is stated in Definition 1.

DEFINITION **1.** *The force between any two particles having masses m_1 and m_2, separated by a distance r, is an attraction acting along the line joining the particles and has the magnitude. Shown in Equation (1).*

$$F = G\frac{m_1 m_2}{r^2} \tag{1}$$

where G is a universal gravitational constant.

The forces between two particles with mass are an action-reaction pair. Two particles with masses m_1 and m_2 exert attracting forces F_{12} and F_{21} towards each other whose magnitudes are equal but their directions are opposed.

The gravitational constant G is an empirical physical constant involved in the computation of the gravitational attraction between particles with mass, which can be determined by the maximum deflection method [11].

$$G = 6.673 \times 10^{-11} N(m/kg)^2 \tag{2}$$

The gravitational force is extremely weak compared to other fundamental forces; e.g. the electromagnetic force is 39 orders of magnitude greater than the gravity force.

Newton's law of universal gravitation can be written in vectorial notation, which considers both: The force of the masses and the direction of each force. The vectorial notation is shown in Equation (3).

$$F_{12} = -G\frac{m_1 m_2}{|r_{12}|^2}\hat{r}_{12} \tag{3}$$

Where F_{12} is the force exerted by m_1 on m_2, G is the gravitational constant, m_1 and m_2 are the masses of the particles, $|r_{12}|$ is the euclidean distance between particles m_1 and m_2, and $\hat{r}_{12}$ is the unit vector, defined as $\frac{r_2-r_1}{|r_2-r_1|}$, r_1 and r_2 are the locations of particles m_1 and m_2. (See Figure 1).

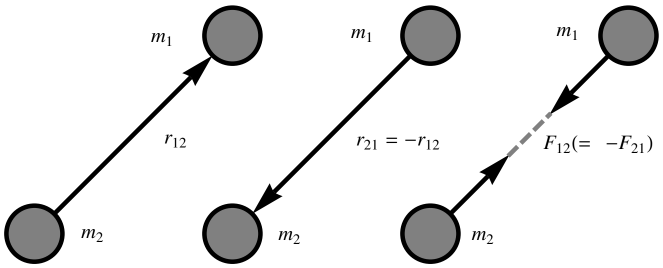


Fig. 1. The force exerted on m_2 (by m_1), F_{21}, is directed opposite to the displacement, r_{12}, of m_2 from m_1. The force exerted on m_1 (by m_2), F_{12}, is directed opposite to the displacement, r_{21}, of m_1 from m_2. $F_{21} = -F_{12}$, the forces being an action-reaction pair.

3 Gravitational Interactions Optimization

In order to find one or more optima there exists a large variety of evolutionary algorithms, e.g. genetics algorithms (GA) [4], evolutionary strategies (ES) [6], ant colony optimization (ACO) [2] , particle swarm optimization (PSO) [8], electrostatic PSO (EPSO) based on electrostatic interactions inspired upon PSO [3], etc. There exist works related to design metaheuristics that take into account the distance in order to determine the cluster membership of the particles computing and maximizing a ratio for all particles in the swarm with respect to the particle to bo updated, e.g. FER-PSO [9]. We propose a Gravitational Interaction Optimization metaheuristic (GIO) capable of solving optimization problems. The motivation of the design of this metaheuristic is to find useful properties and anolgies that can relate optimization problems with Newton's gravitational theory. In the approach presented in this paper, we abduct the interactions exhibited by a set of bodies and use them to guide the search for the global optimum in an optimization problem.

3.1 Gravitational Interactions for Unimodal Optimization

GIO is a population-based metaheuristic where a set of bodies are initially dispersed along the search space with a uniform random distribution. The fitness of bodies located on the search space are mapped as masses in a Gravitational field where the solutions are evolved. Each body stores its current position B and the best position so far B_{best} according to the fitness function. Bodies are allowed to interact in a synchronous discrete manner for a number of epochs. The body interactions follow Newton's gravitational law and move each body to a new location in such way that whole population tends to reach the global optimum (or multiple local optima for multi-modal problems).

The fitness function is a mapping that transforms a vector $X = (x_1, x_2, \ldots, x_n)$ to a scalar $f(X)$. This mapping associates the fitness value $f(X)$ to each location $X = (x_1 \cdots x_n)$ of the search space. We assign a body B to every location X in the search space where an individual of the population is found. Body B is assigned a mass, whose magnitude is a function of the fitness of its location.

Newton's law of universal gravitation describes the attraction forces that exist between two punctual bodies with masses (described in vectorial form in 3). Substituting we obtain Equation (4).

$$\boldsymbol{F_{ij}} = \frac{M\left(f(B_i)\right) \cdot M\left(f(B_j)\right)}{|B_i - B_j|^2} \hat{B_{ij}} \tag{4}$$

Where M is the mapping function that associates the fitness value f of domain $\{x : x \in \Re\}$ a mass of codomain $\{y : y \in (0, 1]\}$ for each position of the body B_i. This mapping is computed using Equation (5).

$$M(f(B_i)) = \left(\frac{f(B_i) - \mathbf{min} f(B)}{\mathbf{max} f(B_{best}) - \mathbf{min} f(B)}(1 - mapMin) + mapMin\right)^2 \tag{5}$$

Where B_i is the position of the ith body and B_j is the jth body that contributes exerting a force on the mass B_i; $|B_i - B_j|$ is the euclidean distance and $\boldsymbol{B_{ij}}$ is the unit vector between bodies B_i and B_j; $f(B_i)$ is the fitness of body B_i, $\mathbf{min} f(B)$ is the minimum fitness value of the current positions of the bodies, $\mathbf{max} f(B_{best})$ is the maximum fitness value of the best positions so far, $mapMin$ is a constant with a small positive value near zero, such that $(1 - mapMin)$ reescales the fitness value $f(B_i)$ to a mass between $[0, 1)$ values. The result is squared to emphasize the best and worst fitnesses.

One characteristic of the proposed method is the full interaction; i.e each body B_i interacts with every other body B_j through their masses. Interactions contribute to their displacement, according to the resultant force. Equation (6) computes the resultant force exerted on body B_i by the bodies B_j.

$$\boldsymbol{F_{ik}} = \sum_{j=1}^{n} \frac{M\left(f(B_i)\right) \cdot M\left(f(B_{j,best})\right)}{|B_i - B_{j,best}|^2} \hat{B_i B_{j,best}} \tag{6}$$

Where $\boldsymbol{F_{ik}}$ is a resultant force of the sum of all vector forces between $M(B_i)$ and $M(B_{j,best})$, $|B_i - B_{best,j}|$ is the Euclidean distance between the current positions of body B_i and the best position so far of the body B_j. In order to avoid numerical errors we compute the force between masses $M(B_i)$ and

$M(P_{j,best})$ only if $|B_i - B_j| \geq \times 10^{-5}$, $B_i\hat{B_{j,best}}$ is the unit vector that directs the force. In order to estimate a displacement that could enhance the solution of particle B_i, it is neccesary to solve Equation (4) for B_j. Assuming that we want to find a location of the body B_k with $M(f(B_k)) = 1$, B_k is computed using Equation (7).

$$B_k = \sqrt{\frac{M(f(B_i))}{|\boldsymbol{F_{ik}}|}}\hat{F_{ik}} \tag{7}$$

To update the position of the bodies we use equation (8) and (9).

$$V_{new} = \chi\left(V + R\cdot C\cdot B_k\right) \tag{8}$$

$$B_{t+1} = B + V_{new} \tag{9}$$

V is the current velocity of B_i, R is a random real number generated in the range of $[0, 1)$ and is multiplied by the gravitational interaction constant C, in order to expect random exploration distances with mean $\mu \approx 1$, we set $C = 2.01$, this displacement is constrained multiplying by a constant with a value of 0.86, in order to ensure the convergence. B_k is the main displacement computed by equation (7).

The complete GIO algorithm is described the Algorithms 1, 2 and 3. Algorithm 1 computes the the total force exerted by the masses $M(f(B_j))$ mass $M(f(B_i))$; in order to prevent premature convergence and division by 0, we compute only those pairs of bodies with a distance greater than ϵ. Algorithm 2 computes the velocities of the bodies, receives the bodies and computes the resultant force that attracts the mass assigned to B_i. In order to prevent a division by 0 we compute the distance only if $|Ftotal| > 0$, the new velocity is computed by Equation (8), and finally we update the velocity associated to B_i. Algorithm 3 computes the new positions B of each iteration t, the algorithm take as parameters the search range, the number of bodies $nBodies$, and the maximum number of iterations $tMax$. The algorithm computes the velocities with $computeVelocities(bodies)$ (Algorithm 2), and updates the their positions with $updatePosition()$, which implements Equation (9), $limitPositions()$ limits the positions of the bodies to the search space defined by the search range, $updateFitness()$ updates the fitness according to the new positions of the bodies and finally we update the best position so far with $updateB_{best}()$.

Algorithm 1. computeFtotal(index)

```
1: $i \leftarrow index$
2: $Ftotal \leftarrow 0$
3: **for** $j \leftarrow 1$ to $nBodies$ **do**
4:     **if** $distance(B_i, P_j) > \epsilon$ **then**
5:         $Ftotal \leftarrow Ftotal + \hat{P_{ij}}M(f(B_i))M(f(P_j))/distance(B_i, P_j)^2$
6:     **end if**
7: **end for**
8: **return** $Ftotal$
```

Algorithm 2. computeVelocities(bodies)

```
1: for $i \leftarrow 1$ to $nBodies$ do
2:     $Ftotal \leftarrow computeFtotal(i)$
3:     if $|Ftotal| > 0$ then
4:         $distance \leftarrow \sqrt{M(f(B_i))}/|Ftotal|$
5:     else
6:         $distance \leftarrow 0$
7:     end if
8:     $V_{new} \leftarrow \chi(V + R\cdot C\cdot distance\cdot \hat{Ftotal})$
9:     $updateVelocity(B_i, V_{new})$
10: end for
11: return $Ftotal$
```

Algorithm 3. MainGravitationalInteraction(ranges, nBodies)

```
1: $bodies \leftarrow initializeParticles(nBodies, ranges)$
2: for $t \leftarrow 0$ to $maxIter$ do
3:     $computeVelocities(bodies)$
4:     $limitVelocity()$
5:     $updatePosition()$
6:     $limitPosition()$
7:     $updateFitness()$
8:     $updateP_{best}()$
9: end for
```

This scheme develops good results for unimodal problems. The results of the performace of the algorithm presented in this Section are presented in Section 4.

3.2 Gravitational Interactions for Multimodal Optimization

In the previous Subsection we showed the basic steps of the gravitational interactions metaheuristic. This scheme works well for unimodal problems. For multimodal problems it is necessary to add a cognitive component analogous to the one used in PSO [8]; the cognitive component is a constant that gives a weight to each body's memory. The new positions of the bodies are computed in order to find more than one optima with the Equations (10) and (11).

Adding the cognitive component to Equation (8) and using the constriction factor χ (Equation 12), makes the new Equation (10) capable to find more than one optimum in multimodal problems. The effect of this component is to make the local search more robust restricting to the bodies to local search, unless the gravitational forces of a cluster of masses overcome the force exerted by its cognitive component.

$$V_{new} = \chi\left(V + C_1\cdot R_1\cdot (B_{best} - B) + C_2\cdot R_2\cdot B_k\right) \tag{10}$$

$$B_{t+1} = B + V_{new} \tag{11}$$

where, analogous to PSO, C_1 and C_2 are the cognitive and the gravitational interaction constants, R_1 and R_2 are real random numbers variables in the $[0, 1)$ range and χ is the inertia constraint (Proposed by Clerk ([8])). The inertia constraint is used to avoid the bodies to explore out of the search space computed by Equation (12).

$$\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \tag{12}$$

where $\phi = C_1 + C_2 > 4$, C_1 and C_2; κ is an arbitrary value in the range of $(0, 1]$. In our algorithm we set $C_1 = C_2 = 2.01$. The constriction factor in our algorithm helps to converge through the iterations. To make multimomodal Gravitational Interactions Algorithm (Algorithms 1, 2, and 3) described in the previous subsection, we replace line 8 in Algorithm 2 by Equation 10.

4 Experiments

In order to test the performance of the Gravitational Interactions Optimization algorithm for unimodal and multimodal functions, we tested both versions with some functions commonly used to measure the performance of different kinds of metaheuristics.

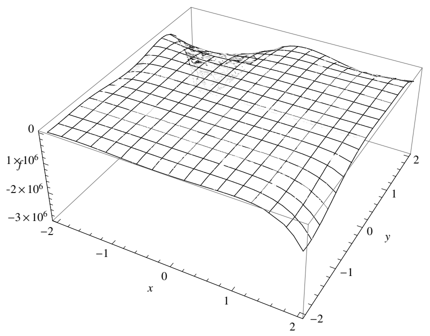
4.1 Test Functions

We show the performance of unimodal and multimodal Gravitational Interactions Optimization algorithm with 3 unimodal and 4 multimodal functions. The test functions tested are shown in the Table 1.

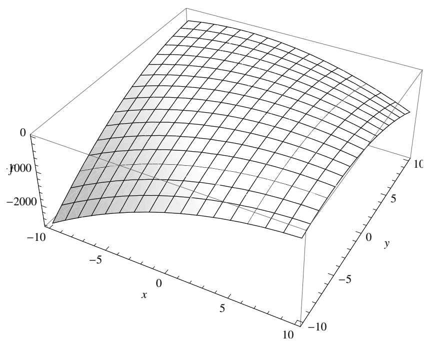
For unimodal optimization we used the functions in Figure 2: $U1$ is the Goldstein and Price function shown in Figure 2(a), $U2$ is the Booth function shown in Figure 2(b) and $U3$ is the 4 variable Colville Function. For multimodal optimization we used the functions of the Figure 3 $M1$ is the Branin's RCOS Function with 3 global optima (with no local optima) shown in Figure 3(a), $M2$ is the

Table 1. Test functions used for our experiments

Unimodal Test Functions		
$U1$	$U1 = [1 + (1 + (x + y + 1)^2)(19 - 14x + 3y^2 + 6xy + 3y^2)] \cdot [(30 + (2x - 3y)^2)(18 - 32x + 12x^2 + 48y - 36xy + 27y^2)]$	$-2 \leq x, y \leq 2$
$U2$	$U2 = (x + 2y - 7)^2 + (2x + y - 5)^2$	$-10 \leq x, y \leq 10$
$U3$	$U3 = -1100 \cdot (w^2 - x)^2 + (w - 1)^2 + (y - 1)^2 + 90 \cdot (y^2 - z)^2 + \cdot\ 10.1 \cdot ((x - 1)^2 + (z - 1)^2) + 19.8 \cdot (x^{-1}) \cdot (z - 1)$	$-10 \leq w, x, y, z \leq 10$
Multimodal Test Functions		
$M1$	$M1 = -\left((y - \frac{5.1x^2}{4\pi^2} + \frac{5x}{\pi} - 6)^2 + 10(1 - \frac{1}{8\pi})Cos(x) + 10\right)$	$-5 \leq x \leq 10$ $0 \leq y \leq 15$
$M2$	$M2 = Sin(5\pi x)^6$	$-0 \leq x \leq 1$
$M3$	$M3 = -(x^2 + y - 11)^2 - (x + y^2 - 7)^2$	$-6 \leq x, y \leq 6$
$M4$	$M4 = -4\left((4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (-4 + 4y^2)y^2\right)$	$-1.9 \leq x \leq 1.9$ $-1.1 \leq x \leq 1.1$

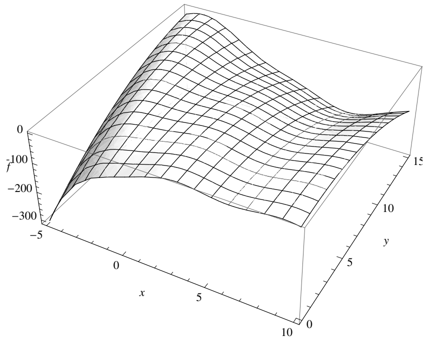


(a) Goldstein-Price function

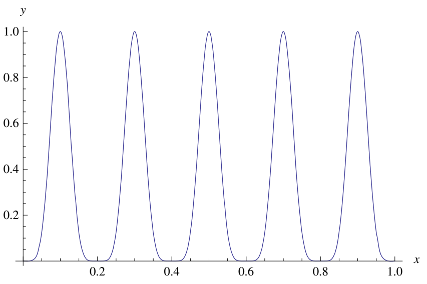


(b) Booth function

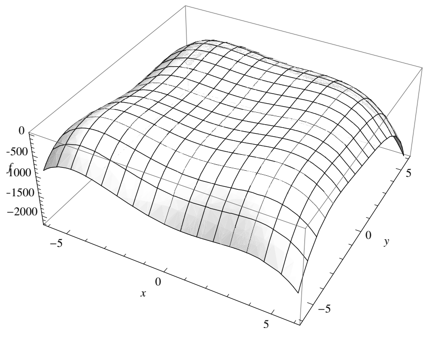
Fig. 2. Fitness landscape of two test functions with one optima used for measure the performance of Unimodal Gravitational Interactions



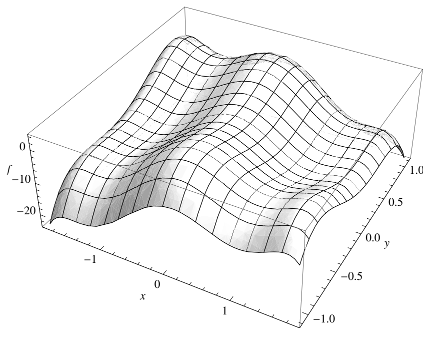
(a) Branin's RCOS function



(b) Deb's function



(c) Himmelblau's function



(d) Six-hump cammelback function

Fig. 3. Fitness landscape of multimodal test functions used in our experiments

6 global maximum univariable Deb's function shown in Figure 3(b), $M3$ is the Himmelblau's function with 4 global optima shown in Figure 3(c), $M4$ is the Six-Hump cammelback function with 2 global optima and 4 local optima shown in Figure 3(d).

4.2 Results

In our experiments we consider $\epsilon = 1 \times 10^{-3}$ to be an acceptable error to determine if the solution obtained had reached the optimum. We used 100 bodies for a maximum of 1000 iterations, we used as stop condition the inability of all the bodies to enhance their fitness memory solutions by 1×10^{-4}, or when the algorithm found all the optima. Each experiment was repeated 30 times. PSO with niches requires two extra parameters: the radius r, and the maximum number of particles on each niche $nMax$, we set $M1$ with $r = 0.5$ and $nMax = 50$, $M2$ with $r = 0.1$ and $nMax = 15$, $M3$ with $r = 0.5$ and $nMax = 30$, and $M4$ with $r = 0.5$ and $nMax = 25$.

The performance of Gravitational Interaction Optimization (GIO) is compared with Particle Swarm Optimization with niches (NPSO) in Table 2, considering the mean and the standard deviation of evaluations required to find all the global optima (column **Evaluations**) and the percentage of successes (column **Success**) to finding all the optima.

Table 2. Results of our experiments

	PSO			**GIO Unimodal**		
Functions	**Evaluations**		Success	**Evaluations**		Success
	μ	σ		μ	σ	
$U1$	1,394.44	399.22	20%	16,523.33	13,928.90	100%
$U2$	1,130.77	330.11	60%	6,057.55	3,984.54	70%
$U3$	764.00	777.75	83%	530.00	208.69	100%
	NPSO			**GIO Multimodal**		
	Evaluations			**Evaluations**		
	μ	σ		μ	σ	
$M1$	2,529.17	764.13	80%	2,803.33	972.90	100%
$M2$	276.66	81.72	100%	390.00	88.44	100%
$M3$	3,400.00	0.00	00.3%	2,323.33	288.496	100%
$M4$	1,136.67	303.41	100%	1,600.00	501.721	100%

The obtained results show that Unimodal and Multimodal Gravitational Interactions have a higher probability to converge to global optima, avoiding premature convergence that PSO and PSO with niches with a similar number of evaluations required for the functions tested.

5 Conclusions

We presented a new heuristic more reliable than PSO with no aditional parameters like the radius r and the maximum number of particles in a niche $nMax$ used in PSO with niches. In problems with high dimentions the radius r is determined by trial and error, because we can not to graph the objective function and make a visual analysis. The same algorithm is used for unimodal and multimodal cases. When used in its general form. (i.e. including the cognitive component), GIO solves both cases without the need of any a-priori information. Adding the cognitive component allow us to solve both, unimodal and multimodal optimization problems, while GSA can only solve unimodal problems. While GIO has proven to find all optima in a multimodal problem, GSA can only determine one of them.

References

1. Stretching technique for obtain global minimizers through particle swarm optimization. In: Proceedings of the Particle Swarm Optimization Workshop (2001)
2. Dorigo, M., Maniezzo, V., Colorni, A.: Distributed Optimization by Ant Colonies. Elsevier Publishing, Amsterdam (1992)
3. Barrera, J., Coello, C.A.C.: A particle swarm optimization method for multimodal optimization based on electrostatic interaction (2009)
4. Goldberg David, E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Pub. Co., Reading (1989)
5. Rashedi, E., Nezamabadi-pour, H., Saryazdi, S.: GSA: A Gravitational Search Algorithm (2009)
6. Ingo, R.: Evolutionsstrategie 1994. Frommann Holzboog (1994)
7. Plagianakos, V.P., Magoulas, G.M., Parsopoulos, K.E., Vrahatis, M.N.: Improving particle swarm optimizer by function stretching. Nonconvex Optimization and Applications 54
8. Kennedy, J., Eberhart, R.: Swarm Intelligence. Evolutionary Computation. Morgan Kaufmann Publisher, San Francisco (2001)
9. Li, X.: A multimodal particle swarm optimizer based on fitness euclidean-distance ratio. In: Proceedings of the 9t annual conference on Genetin and evolutionary computation (GECCO 2007), pp. 78–85 (2007)
10. Newton, I.: Newtons Principia Mathematica. Fsica. Ediciones Altaya, S.A., 21 edition (1968)
11. Robert, H., David, R.: Physics Part I. Physics. John Wiley and Sons Inc., Chichester (1966)

Particle Swarm Optimization with Resets – Improving the Balance between Exploration and Exploitation

Yenny Noa Vargas[1] and Stephen Chen[2]

[1] Department of Artificial Intelligence and Computational Systems, University of Havana, Havana, Cuba
yenny@matcom.uh.cu
[2] School of Information Technology, York University, Toronto, ON M3J 1P3 Canada
Phone: 416-736-2100 x: 30526; Fax: 416-736-5287
sychen@yorku.ca

Abstract. Exploration and exploitation are two important factors to consider in the design of optimization techniques. Two new techniques are introduced for particle swarm optimization: "resets" increase exploitation and "delayed updates" increase exploration. In general, the added exploitation with resets helps more with the lbest topology which is more explorative, and the added exploration with delayed updates helps more with the gbest topology which is more exploitive.

Keywords: Particle Swarm Optimization, Search Intensification, Search Diversification.

1 Introduction

To perform an effective balance between exploration and exploitation, two tasks must be taken into account: quickly identify regions in the search space with high quality solutions without wasting too much time in regions which are either already explored or which do not provide high quality solutions, and perform an intense search exploiting the collected search experience to locate the optimal solutions. These two tasks are conflicting and equally important so a trade-off between these two objectives – exploration and exploitation – must be achieved.

Particle swarm optimization (PSO) [1] is a simple search technique that has shown excellent search abilities and good results in several optimization problems. Like many search techniques, standard PSO can not easily solve multi-modal optimization problems because of the lack of an explicit strategy to escape from local optima. This is usually faced as a problem of the balance between exploration and exploitation of the search space. Consequently, several techniques have been proposed to accomplish an effective trade-off between these two search components [2] [3] [4].

This paper presents and empirically analyses various forms of *particle swarm optimization with resets* with the aim of improving the balance between exploration and exploitation. The *extreme resets* and *resets allowing exploration* strategies are inspired by some well recognized metaheuristic techniques like intensification in Tabu

Search (TS) [5] [6] [7]. *Delayed update* is applied as a diversification strategy with its effectiveness caused by its prolongation on the exploration phase.

Results for a set of multi-modal functions show that the global communication (*gbest*) and local communication (*lbest*) topologies of PSO can both benefit from resets and delayed updates. In general, resets (which increase exploitation) are more effective with lbest, and delayed updates (which increase exploration) are more effective with gbest. Conversely, resets are generally ineffective with gbest, and the benefits of delayed updates degrade more quickly with lbest.

This paper begins in section II with an introduction of relevant aspects that were considered to propose the resets and delayed updates strategies in section III. Experiments with these techniques with the gbest and lbest models of PSO are performed on the multi-modal set of BBOB problems and their results are analyzed in section IV before a summary is provided in section V.

2 Background

The balance between exploration and exploitation is not new in the metaheuristic community (e.g. VNS, ILS, ACO) [8] [9] [10] [11]. Some metaheuristics can be seen as "intelligent" extensions of local search whose main objective is to escape from local optima to continue exploration of the search space to find other local optima and hopefully the global optimum. This is the case of Tabu Search (TS) [5] [6] [7] which combines local search with the use of choice rules to promote the search process to examine neighbors of elite solutions historically found (*intensification*) and unvisited regions or solutions that differ in various significant ways from those seen before (*diversification*).

Other techniques benefit from a natural way to explore the search space by using a set of solutions rather than a single solution (e.g. Evolutionary Algorithms (EAs) [12][13][14]). To complement the intrinsic ability of (population-based) exploration with an exploitation approach, EAs use recombination operators of solutions with good features to hopefully get better solutions (*exploitation*) and mutation operators to diversify the search (*exploration*).

Particle Swarm Optimization (PSO) [1][15] is related to EAs in the sense of using more than one solution to explore the search space, but unlike these algorithms it uses attractor coefficients towards high-quality solutions in its neighborhood leading the search to better areas and hopefully the global solution.

Two popular communication topologies in PSO are the *global topology* or *gbest model* and the *local topology* or *lbest model*. In the first, the neighborhood of each particle is the whole swarm, so any member of the swarm benefits from what every other particle has learned. The second refers to any swarm model without global communication. The simplest form of a local topology is the ring model [16] where each particle is connected to only two particles in the swarm.

As reported in the literature [15] [16], the gbest model has the advantage of fast convergence while the lbest model benefits from a longer exploration process. The fast convergence of the gbest model makes it vulnerable to premature convergence into local optima. On the contrary, the lbest model should explore in parallel different areas of the search space and its partial communication makes it less likely to converge prematurely to local optima. Experimentally, this slower convergence leads to better overall performance [16]. In this sense, the two models can be characterized as more exploitive (gbest) and more explorative (lbest).

A simple method to improve the balance between exploration and exploitation in PSO might be to use a topology in between the two extremes of the gbest and lbest topologies since the main difference between these two models is the social network the particles use to communicate. A logical procedure is to make experiments varying the numbers of neighbors of each particle to find a suitable value. Unfortunately, this idea results in a multi-objective problem of parameter optimization in which there is no single solution but a front of Pareto Optimal solutions [17]. These solutions have the characteristic that any one of them could be said the best solution because they are *non-dominated* solutions. As the communication network gets closer to the gbest model, convergence is faster but at the cost of to increase its vulnerability to stagnate in local optima. Conversely, if the solution approaches the lbest model, its capability of exploration is higher but at the cost of slower convergence.

Another approach might be to vary from a more explorative to a more exploitive process while searching with the aim of tuning the balance between exploration and exploitation. This idea has been successfully performed in other search techniques like Simulated Annealing (SA) [18] [19] through its cooling rule – at the beginning of the search the temperature might be a greater value in order to sample the search space and gradually could decrease, following a specific rule, to converge into a local minimum at the end of the search. Another variant derived from the above is to alternate phases of this process (e.g. cooling and reheating in SA) and thus provide an oscillating balance between exploration and exploitation [20]. A problem with these approaches is the difficulty in selecting an effective form in which the algorithm will turn from more explorative to more exploitive phases, and to determine what is the best moment to begin with another exploration-exploitation phase.

To avoid these problems many algorithms go for another way to balance the exploration and exploitation of the search space: they use a *restart* mechanism or multiple local optimizations like Memetic Algorithms [21]. The idea is very simple: perform a local search until certain conditions become satisfied (i.e. stagnation in local optimum, no diversity, etc.) and then restart the algorithm in regions not visited. Simple restart is a "blind" process which is not guaranteed to search in promising unexplored regions.

The next section presents some explicit strategies to improve the balance between exploration and exploitation in PSO. These strategies are influenced by the ideas discussed in this section and especially in the intensification phase of TS [5] [6] [7].

3 Particle Swarm Optimization with Resets and Delayed Updates

Resets. Resets are a simple way of concentrating the search effort in the most promising areas of the search space. The process builds from the standard PSO [16] by adding a search intensification phase performed every *k* iterations – the particles are reset to the best positions individually collected during the last *k* iterations (which can be the initial point which would be the best position found in the previous *k* iterations). Resets are different from restarts in that no other parameters of the swarm are reset (e.g. to their initial values). Like the intensification strategy in Tabu Search [5] [6] [7], a reset causes the swarm to return to promising regions to search them more thoroughly. The implementation requires storing the best solutions using an explicit memory – *pkbest* – to remember where they have had prior success. Two forms of resets arise from this procedure: the "resets" (R-PSO) and "resets allowing exploration" (RE-PSO). The outline of these procedures is as follows:

PSO with Resets (R-PSO). *nbest*$_i$ could be *gbest* or *lbest* depending on the topology used.

```
for each time step t
  for each particle i do
   update velocity $v_i^t$ and position $x_i^t$ using equations
     $v_i^{t+1} = wv_i^t + c_1r_1(pbest_i^t - x_i^t) + c_2r_2(nbest_i^t - x_i^t)$
     $x_i^{t+1} = x_i^t + v_i^{t+1}$
   update $pbest_i^t$ and $nbest_i^t$ if necessary
  end
  if k iterations is reached then
   for each particle i do
     reset its position:
      $x_i^t = pbest_i^t$
  end
end
```

PSO with Resets allowing Exploration (RE-PSO). *nbest*$_i$ could be *gbest* or *lbest* depending on the topology used.

```
for each particle i do
  $pkbest_i = pbest_i$
end
for each time step t
 for each particle i do
  update velocity $v_i^t$ and position $x_i^t$ using equations
      $v_i^{t+1} = wv_i^t + c_1r_1(pbest_i^t - x_i^t) + c_2r_2(nbest_i^t - x_i^t)$
      $x_i^{t+1} = x_i^t + v_i^{t+1}$
   update $pbest_i^t$ and $nbest_i^t$ if necessary
  end
  if k iterations is reached then
   for each particle i do
    if $f(pbest_i^t) < f(pkbest_i)$ then
      $x_i^t = pbest_i^t$
    end
    $pkbest_i = pbest_i^t$
   end
  end
 end
```

R-PSO performs the resets for each of the particles without any conditions. More specifically, once the amount of *k* iterations is reached, all the particles go back to their *pbest*. In contrast, RE-PSO only performs resets on the particles which have experienced an improvement in the past *k* iterations; otherwise the particles continue on their way. The difference between these resets procedures, in respect to the balance of exploration and exploitation, is that RE-PSO should be more explorative than R-PSO. If the particles don’t improve their *pbest*, it is possible that they are converged into a local optima. In this case, the logical movement is to search in other regions of the search space.

Delayed Updates. Another method to alter the balance between exploration and exploitation is with “delayed updates”. This mechanism updates the *pbest* and *gbest* positions not in real time, but every *k* iterations. The effectiveness of this kind of update is caused by its prolongation on the exploration phase (decrement of the rate of convergence). The isolation of the particles (through incommunication) will permit a broader exploration of the search space around previous (local) best positions and in consequence also will produce a decrement of the rate of convergence in direct proportion to the value of *k*.

The implementation of this procedure can be done easily, and it only requires an extra variable to save the improvements of each particle during *k* iterations. In essence, each particle will have two local memories: *pkbest* to store its best position found in the current period of *k* iterations and *pbest* to guide its movements during these *k* iterations (which will be the previous *pkbest*). The update process of this procedure is as follows:

PSO with Delayed Updates (DU-PSO). *nbest*$_i$ could be *gbest* or *lbest* depending on the topology used.

```
for each particle i do
  $pkbest_i = pbest_i$
end
for each time step t
  for each particle i do
   update velocity $v_i^t$ and position $x_i^t$ using equations
      $v_i^{t+1} = wv_i^t + c_1r_1(pbest_i - x_i^t) + c_2r_2(nbest_i - x_i^t)$
      $x_i^{t+1} = x_i^t + v_i^{t+1}$
   update $pkbest_i$ if necessary
  end
  if k iterations is reached then
   for each particle i do
    if $f(pkbest_i) < f(pbest_i)$ then
      $pbest_i = pkbest_i$
    end
    update $nbest_i$ if necessary
   end
  end
end
```

The delayed updates mechanism can be easily combined with the two forms of resets. Both mechanisms are performed every *k* iterations, thus the same moment in which one of them is performed can be used to perform the other. The implementation of this can be done based in R-PSO and RE-PSO procedures, the difference stays in that the update of *gbest* is done after the update of *pbest* and the resets are performed. The process of this procedure is as follows:

PSO with Resets and Delayed Updates (RDU-PSO). *nbest$_i$* could be *gbest* or *lbest* depending on the topology used.

```
for each particle i do
  $pkbest_i$ = $pbest_i$
end
for each time step t
  for each particle i do
   update velocity $v_i^t$ and position $x_i^t$ using equations
     $v_i^{t+1} = wv_i^t + c_1r_1(pbest_i - x_i^t) + c_2r_2(nbest_i - x_i^t)$
     $x_i^{t+1} = x_i^t + v_i^{t+1}$
   update $pkbest_i$ if necessary
  end
  if k iterations is reached then
   for each particle i do
    if $f(pkbest_i) < f(pbest_i)$ then
      $pbest_i$ = $pkbest_i$
    end
    reset its position:
      $x_i^{t+1}$ = $pbest_i$
    update $nbest_i$ if necessary
   end
  end
end
```

4 Analysis of Results

To analyze the effects of the “resets” and the “delayed updates” mechanisms in gbest and lbest models of PSO, their implementations were applied to a subset of the Black-Box Optimization Benchmarking (BBOB) problems [22] for $D = 20$ and with $k = 1$, 10, 50, 100 and 1000. The BBOB problems come in five sets which are 1–separable functions, 2–functions with low or moderate conditioning, 3–unimodal functions with high conditioning, 4–multi-modal functions with adequate global structure, and 5–multi-modal functions with weak global structure. The subset of “multimodal functions with adequate global structure” was selected for the experiments because their search spaces reward both exploration (multimodality) and exploitation (global structure).

Table 1. Names of the “multimodal functions with adequate global structure” in the BBOB problem set

	Function Name
15	Rastrigin (rotated)
16	Weierstrass
17	Schaffers F7
18	Schaffers F7, moderated ill-conditioned
19	Composite Griewank-Rosenbrock

Using the first five instances for each of these functions, five trials for each instance were run for a total of 25 trials per function. The mean errors for these experiments are reported in Tables 2-3.

Table 2. BBOB Results for gbest PSO with Resets (R-PSO$_G$) for D = 20. Mean error from known optimum for R-PSO$_G$ and gbest PSO with "resets allowing exploration" (RE-PSO$_G$) on functions 15-19 of BBOB problem set with D = 20. For each value of k = 1, 10, 50, 100 and 1000, 25 total trials were run – five independent trials on each of the first five instances of each BBOB function. Values in bold represent improvements of these algorithms over PSO$_G$ – a benchmark gbest PSO [23].

Algorithm	Function	*k*=1	*k*=10	*k*=50	*k*=100	*k*=1000	PSO$_G$
R-PSO$_G$	15	3.61E+02	6.63E+01	6.18E+01	**5.64E+01**	6.16E+01	5.89E+01
	16	1.76E+01	5.91E+00	5.29E+00	6.15E+00	**5.24E+00**	5.25E+00
	17	7.63E+00	1.41E+00	1.59E+00	1.50E+00	1.42E+00	1.15E+00
	18	2.80E+01	4.43E+00	4.84E+00	5.21E+00	5.07E+00	3.65E+00
	19	1.13E+01	**3.83E+00**	**3.92E+00**	**3.89E+00**	4.06E+00	3.98E+00
RE-PSO$_G$	15	**5.66E+01**	6.87E+01	6.39E+01	6.48E+01	**5.76E+01**	5.89E+01
	16	**4.87E+00**	6.38E+00	5.88E+00	**5.19E+00**	**5.16E+00**	5.25E+00
	17	1.35E+00	1.51E+00	1.29E+00	1.26E+00	1.28E+00	1.15E+00
	18	3.80E+00	5.87E+00	4.19E+00	5.14E+00	4.63E+00	3.65E+00
	19	**3.86E+00**	**3.85E+00**	3.98E+00	4.00E+00	3.98E+00	3.98E+00

Table 3. BBOB Results for lbest PSO with Resets (R-PSO$_L$) for D = 20. Mean error from known optimum for R-PSO$_L$ and lbest PSO with "resets allowing exploration" (RE-PSO$_L$) on functions 15-19 of BBOB problem set with D = 20. For each value of k = 1, 10, 50, 100 and 1000, 25 total trials were run – five independent trials on each of the first five instances of each BBOB function. Values in bold represent improvements of these algorithms over PSO$_L$ – an implementation based on the benchmark gbest PSO [23].

Algorithm	Function	*k*=1	*k*=10	*k*=50	*k*=100	*k*=1000	PSO$_L$
R-PSO$_L$	15	3.68E+02	**4.86E+01**	**5.23E+01**	**5.40E+01**	**5.46E+01**	5.59E+01
	16	1.96E+01	6.16E+00	6.00E+00	6.06E+00	**5.45E+00**	5.45E+00
	17	7.20E+00	7.56E-01	6.93E-01	7.29E-01	7.69E-01	6.80E-01
	18	2.61E+01	**3.11E+00**	**2.94E+00**	**3.21E+00**	**3.31E+00**	3.55E+00
	19	1.12E+01	3.67E+00	**3.63E+00**	**3.44E+00**	**3.55E+00**	3.67E+00
RE-PSO$_L$	15	5.64E+01	**4.95E+01**	6.30E+01	**5.21E+01**	5.94E+01	5.59E+01
	16	**5.38E+00**	5.98E+00	5.70E+00	5.94E+00	**5.25E+00**	5.45E+00
	17	7.00E-01	7.27E-01	7.44E-01	7.47E-01	**6.74E-01**	6.80E-01
	18	**2.98E+00**	**3.33E+00**	**3.10E+00**	**3.24E+00**	**2.78E+00**	3.55E+00
	19	**3.58E+00**	3.67E+00	3.79E+00	**3.61E+00**	3.74E+00	3.67E+00

Analyzing the results in Tables 2-3, the following conclusions can be stated:

1. Resets (which increase exploitation) provide little value to gbest models which are already exploitive enough.
2. The resets allowing exploration mechanism produces better results than the resets mechanism.
3. The improvements are more significant in the lbest model than the gbest model.

All this reconfirms the explorative character of the lbest model as well as the exploitive nature of the gbest model. As previously said, the two reset procedures are intensification mechanisms that turn both PSOs into a more exploitive search technique. Therefore, the lbest model exploits the benefits of these mechanisms more than the gbest model. Furthermore, the functions used in the experiments are multimodal, so they benefit from techniques with large capability to explore. Thus, the results get better when the resets are less extreme (R-PSO) but are combined with some exploration (RE-PSO).

While the intensification mechanisms encourage the exploitation of the search space favoring the lbest model, the stimulation of the exploration with "delayed updates" gives more benefit to the gbest model (see Tables 4-5). To solve multimodal functions, the search strategy should be capable of escaping from local optima, so it needs an efficient mechanism of exploration. The Tables 4-5 show that the gbest model improves in performance on all functions with delayed updates for *k* = 10 and *k* = 50. Conversely, the benefits of delayed updates diminish more quickly with the lbest model and there are few improvements for *k* = 100. For both gbest and lbest models, *k* = 1000 leads to a search procedure that is too exploratory to converge to good solutions in reasonable time.

Table 4. BBOB Results for gbest PSO with Resets & Delayed Updates (RDU-PSO$_G$) for D = 20. Mean error from known optimum for gbest PSO with "delayed updates" (DU-PSO$_G$), RDU-PSO$_G$ and RE-PSO$_G$ with "delayed updates" (REDU-PSO$_G$) on functions 15-19 of BBOB problem set with D = 20. For each value of k = 1, 10, 50, 100 and 1000, 25 total trials were run – five independent trials on each of the first five instances of each BBOB function. Values in bold represent improvements of these algorithms over PSO$_G$.

Algorithm	Function	*k*=1	*k*=10	*k*=50	*k*=100	*k*=1000	PSO$_G$
DU-PSO$_G$	15	6.25E+01	**5.14E+01**	**5.32E+01**	6.39E+01	1.71E+02	5.89E+01
	16	**4.95E+00**	**4.17E+00**	**4.35E+00**	**5.07E+00**	1.45E+01	5.25E+00
	17	**1.14E+00**	**4.94E-01**	**5.72E-01**	**4.64E-01**	3.27E+00	1.15E+00
	18	5.14E+00	**2.65E+00**	**3.35E+00**	**2.59E+00**	1.19E+01	3.65E+00
	19	**3.85E+00**	**3.89E+00**	**3.83E+00**	**3.82E+00**	5.15E+00	3.98E+00
RDU-PSO$_G$	15	3.38E+02	**5.61E+01**	**5.33E+01**	6.70E+01	1.72E+02	5.89E+01
	16	1.91E+01	**5.21E+00**	**4.27E+00**	**4.24E+00**	1.52E+01	5.25E+00
	17	7.38E+00	**1.08E+00**	**8.83E-01**	**5.33E-01**	3.23E+00	1.15E+00
	18	2.48E+01	**3.29E+00**	**2.78E+00**	**2.79E+00**	1.16E+01	3.65E+00
	19	1.15E+01	**3.61E+00**	**3.87E+00**	3.99E+00	5.38E+00	3.98E+00
REDU-PSO$_G$	15	6.34E+01	**5.03E+01**	**5.07E+01**	6.27E+01	1.63E+02	5.89E+01
	16	5.77E+00	5.45E+00	**4.37E+00**	**4.12E+00**	1.51E+01	5.25E+00
	17	1.18E+00	**8.19E-01**	**6.18E-01**	**6.69E-01**	3.42E+00	1.15E+00
	18	4.71E+00	**2.26E+00**	**2.67E+00**	**2.03E+00**	1.13E+01	3.65E+00
	19	**3.89E+00**	**3.94E+00**	4.13E+00	3.98E+00	5.07E+00	3.98E+00

As results in Tables 4-5 show, both models benefit from delayed updates a little more than with its combination with any form of resets. In the gbest model, this is because the use of resets causes more exploitation, and the gbest model is already exploitive enough. In the lbest model, the cause seems to be related with an excessively slow convergence. Delaying updates decreases the rate of convergence, which is another reason why the gbest model benefited more from this strategy. The combination of delayed updates with resets, which is another form to slow the convergence (with the return every *k* iterations to visited regions), makes the search process in the lbest model more exploitive but with limited possibilities of convergence.

Table 5. BBOB Results for lbest PSO with Resets & Delayed Updates (RDU-PSO$_L$) for D = 20. Mean error from known optimum for lbest PSO with "delayed updates" (DU-PSO$_L$), RDU-PSO$_L$ and RE-PSO$_L$ with "delayed updates" (REDU-PSO$_L$) on functions 15-19 of BBOB problem set with D = 20. For each value of k = 1, 10, 50, 100 and 1000, 25 total trials were run – five independent trials on each of the first five instances of each BBOB function. Values in bold represent improvements of these algorithms over PSO$_L$.

Algorithm	Function	*k*=1	*k*=10	*k*=50	*k*=100	*k*=1000	PSO$_L$
	15	**5.45E+01**	**4.66E+01**	6.55E+01	8.06E+01	2.16E+02	5.59E+01
	16	**5.28E+00**	**4.79E+00**	**5.36E+00**	6.11E+00	1.35E+01	5.45E+00
DU-PSO$_L$	17	7.83E-01	**5.13E-01**	**5.93E-01**	7.48E-01	4.66E+00	6.80E-01
	18	**3.30E+00**	**2.43E+00**	**2.68E+00**	**3.24E+00**	1.57E+01	3.55E+00
	19	3.72E+00	**3.40E+00**	**3.65E+00**	3.70E+00	5.82E+00	3.67E+00
	15	3.88E+02	**5.11E+01**	6.99E+01	7.97E+01	2.21E+02	5.59E+01
	16	1.95E+01	**4.96E+00**	**4.90E+00**	6.28E+00	1.56E+01	5.45E+00
RDU-PSO$_L$	17	7.01E+00	**4.76E-01**	**5.33E-01**	7.53E-01	4.66E+00	6.80E-01
	18	2.77E+01	**2.72E+00**	**2.41E+00**	**2.86E+00**	1.66E+01	3.55E+00
	19	1.10E+01	3.72E+00	3.87E+00	3.97E+00	5.62E+00	3.67E+00
	15	5.80E+01	5.76E+01	6.77E+01	8.10E+01	2.21E+02	5.59E+01
	16	5.51E+00	**5.36E+00**	**5.15E+00**	6.94E+00	1.44E+01	5.45E+00
REDU-PSO$_L$	17	**6.01E-01**	**5.08E-01**	**5.70E-01**	7.89E-01	4.43E+00	6.80E-01
	18	**3.17E+00**	**2.82E+00**	**2.94E+00**	**3.01E+00**	1.64E+01	3.55E+00
	19	3.76E+00	3.70E+00	3.95E+00	3.92E+00	5.79E+00	3.67E+00

In parallel implementations, the communication overhead is very important because it influences the use of distributed and shared resources. As lbest tends to perform better than gbest and it uses less communication overhead, parallel implementations of PSO often use the lbest model. The above results suggest that the use of delayed updates may be a useful technique to try with parallel implementations of particle swarm optimization.

5 Summary

In order to improve the trade-off between exploration and exploitation in the search process of particle swarm optimization, the "resets" and "delayed updates" techniques have been introduced. Different PSO implementations that combine these techniques with the gbest and lbest models have been evaluated by using the multi-modal BBOB problems. The results show that with these new techniques, an improved balance between exploration and exploitation can be achieved on both models. Applying these insights to parallel implementations of PSO is a promising area for future research.

References

1. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948. IEEE Service Center, Piscataway (1995)
2. Cartwright, L., Hendtlass, T.: A Heterogeneous Particle Swarm. In: Korb, K., Randall, M., Hendtlass, T. (eds.) Proceedings of Fourth Australian Conference on Artificial Life, pp. 201–210. Springer, Heidelberg (2009)
3. Chen, S.: Locust Swarms – A New Multi-Optima Search Technique. In: Proceedings of the 2009 IEEE Congress on Evolutionary Computation, pp. 1745–1752 (2009)
4. Hendtlass, T.: WoSP: A Multi-Optima Particle Swarm Algorithm. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, pp. 727–734 (2005)
5. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Dordrecht (1997)
6. Glover, F.: Tabu Search. ORSA Journal on Computing 1, 190–206 (1989)
7. Glover, F.: Tabu Search Part II. ORSA Journal on Computing 2(1), 4–32 (1990)
8. Hansen, P., Mladenović, N.: An Introduction to variable neighborhood search. In: Voß, S., Martello, S., Osman, I., Roucairol, C. (eds.) Methaheuristics: Advances and trends in local search paradigms for optimization, ch.30, pp. 433–458. Kluwer Academic Publishers, Dordrecht (1999)
9. Stützle, T.: Iterated local search for the quadratic assignment problem. Technical report, aida-99-03, FG Intellektik, TU Darmstadt (1999)
10. Lourenço, H.R., Martin, O., Stützle, T.: A beginners̀ introduction to Iterated Local Search. In: Proceedings of MIC 2001 Metaheuristics International Conference, Porto, Portugal, vol. 1, pp. 1–6 (2001)
11. Dorigo, M., Gambardella, L.: Ant Colony System: a cooperative learning approach to the traveling salesman problem. IEEE Transaction on Evolutionary Computation 1, 53–66 (1997)
12. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading (1989)
13. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge (1998)
14. Rechenberg, I.: Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (PhD thesis). Fromman-Holzboog (1973)
15. Eberhart, R.C., Kennedy, J.: A New Optimizer using Particle Swarm Theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39–43. IEEE Service Center, Piscataway (1995)
16. Bratton, D., Kennedy, J.: Defining a Standard for Particle Swarm Optimization. In: Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007), pp. 120–127 (2007)
17. Kalyanmoy, D.: Multi-Objective Optimization using Evolutionary Algorithms. Department of Mechanical Engineering. Institute of Technology, Kanpur, India (2001)
18. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
19. Aarts, E.H.L., Korst, J.H.M., Laarhoven, P.J.M.: Simulated Annealing. In: Local Search in Combinatorial Optimization. In: Aarts, E.H.L., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 91–120. Wiley Interscience, Chichester (1997)
20. Blum, C., Roli, A.: Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison

21. Moscato, P., Cotta, C.: An Introduction to Memetic Algorithms. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 19, 131–148 (2003)
22. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. INRIA Technical Report RR-6829 (2009)
23. El-Abd, M., Kamel, M.S.: Black-Box Optimization Benchmarking for Noiseless Function Testbed using Particle Swarm Optimization. In: Proceedings of the 2009 Genetic and Evolutionary Computation Conference, pp. 2269–2273 (2009)

MiTS: A New Approach of Tabu Search for Constructing Mixed Covering Arrays

Loreto Gonzalez-Hernandez and Jose Torres-Jimenez

Information Technology Laboratory, CINVESTAV-Tamaulipas
Km. 5.5 Carretera Cd. Victoria-Soto la Marina, 87130, Cd. Victoria Tamps., Mexico
agonzalez@tamps.cinvestav.mx, jtj@cinvestav.mx
http://www.tamps.cinvestav.mx/~jtj/

Abstract. Software systems have been increasingly used by our society, so a failure in them can lead to large losses. To reduce the failures of a software it is necessary to carry out the testing process appropriately. The combinatorial testing helps in the testing process by providing structures with a test set of small size, like Mixed Covering Arrays (MCAs). However, the problem of constructing an optimal test set is an NP-complete problem leading to the development of non exhaustive approaches to solve it. This paper proposes a new approach of Tabu Search (TS) called MiTS (that stands for Mixed Tabu Search) which focuses on constructing MCAs. The approach is based on the use of a mixture of neighborhood functions and a fine tuning process to improve the performance of the TS. Experimental evidence shows a poor performance when a single neighborhood function is used. In the other side, the TS (using a mixture of neighborhood functions) is competitive in the construction of MCAs over a known benchmark reported in the literature.

Keywords: Mixed Covering Arrays, software testing, Tabu Search.

1 Introduction

Software systems have been increasingly used by our society, so a failure in them can lead to large losses [20]. For instance, the National Institute of Standards and Technology (NIST) [23] reports that software defects affect to the USA economy with close to \$60 billion per year and assumes that approximately \$22 billion could be reduced through more effective testing techniques, in this regards, Hartman [12] mentions that the quality of software is intrinsically related to the appropriate testing techniques used to deliver it. Based on this fact, we see the importance of an adequate testing process. Despite these advantages, Hinch et. al. [15] comment that sometimes this stage consumes more than the half of the total cost of the development, while Hartman [13] affirms that it is more than 80% so, to carry out the verification in an exhaustive way, most of the times is outside of a reasonable time and budget. For instance, suppose that a system has 12 parameters each with 4 possible values, for testing all the combinations it is necessary to use $4^{12} = 16,777,216$ test cases, this quantity is too large for practical purposes. In the same way, when the number of values of each parameter increases, also the total number of configurations grows exponentially, for this reason, another alternative has to be taken in order to conduct the tests using the minimum of possible cases and the maximum coverage.

Recent studies have shown that close to 100% of the known failures of systems of various kinds are exhibited using interactions of size 6 (i.e. taking all the parameters combinations of size 6) [17], this means that it is possible to construct an effective test set, if it contains all the interactions of size 6 of the parameters of a system, in this way, the overall number of test cases can be reduced significantly. Under this criterion, the example given previously would require $14,288$ tests [7]. The combinatorial approach is an acceptable approach that influences in the cost of the tests and the required degree of coverage [33]; this exposition has even been used in the reduction of costs in the industry. This approach uses combinatorial structures to represent the test cases, the most common ones are Orthogonal Arrays (OAs), Covering Arrays (CAs) and Mixed Covering Arrays (MCAs). In general, OAs, CAs and MCAs are matrices where each column contains the values of each parameter and a row indicates the combination of each of them to construct a test. When the matrix has the minimum possible number of rows is called optimal.

The first work that used combinatorial structures in a practical application was presented in 1926 by Fisher [9]. Fisher applied all the possible combinations for each pair of factors, since then the tests of controlled interaction have been used. In addition to the above, Hedayat, et. al. [14] reported that combinatorial models are mainly used for the design of experiments, being important in all areas of human research, e.g. in medicine, agriculture and manufacturing.

The main problem of interaction testing focuses on constructing a test set which contains the coverage indicated with the minimum number of tests. In [19] and [8] the NP completeness of the problem of constructing CA was reported.

Due to the complexity of the problem, different approaches have been implemented to provide good solutions in a reasonable time. Some of these techniques are: greedy algorithms [29], genetic algorithms (GA) [26], tabu search (TS) [22], simulated annealing (SA) [6], ant colony optimization algorithm (ACO) [25], hill climbing (HC) [6], great deluge (GD) [3], SAT model [21] among others. Previous TS implementations used a single neighborhood function to move from one solution s to another solution s'. The present paper presents the construction of MCA which uses a mixture of neighborhood functions.

This paper is organized as follows: the description of combinatorial testing and OA, CA and MCA are given in section 2. Some techniques that have been used for constructing CA are presented in section 3. An overview of the proposed approach of TS with the details of its main components and values are described in section 4. Then, the design of the experiment and computational results are shown in section 5 and 6 respectively. Last section summarizes the main contribution of this paper.

2 Mixed Covering Arrays

If a software system has k parameters each with v possible values, it would take v^k configurations if it wants to be tested in an exhaustive way, however another method called interaction testing can be used. The main goal of this approach is to find the minimum possible number of tests which satisfies that all the interactions of a predefined size are covered. The combinatorial structures used for interaction testing are mainly orthogonal arrays and covering arrays [8].

An orthogonal array (OA), denoted by $\mathrm{OA}_\lambda(N; t, k, v)$, is an $N \times k$ array on v symbols such that every $N \times t$ sub-array contains all the ordered subsets of size t from v symbols exactly λ times. Orthogonal arrays have the property that $\lambda = \frac{N}{v^t}$. When $\lambda = 1$ it can be omitted from the notation and the OA is optimal. An orthogonal array $\mathrm{OA}(N; t, k, v)$ is a special type of $\mathrm{CA}(N; t, k, v)$.

A covering array (CA) is a mathematical object, denoted by $\mathrm{CA}(N; k, v, t)$ which can be described like a matrix with $N \times k$ elements, such that every $N \times t$ sub-array contains all possible combinations of v^t symbols at least once. N represents the rows of the matrix, k is the number of parameters, which has v possible values and t represents the strength or the degree of controlled interaction. A CA has the same cardinality in all their parameters. However, software systems are generally composed with parameters that have different cardinalities; in this situation a mixed covering array (MCA) can be used.

A mixed covering array, denoted by $\mathrm{MCA}(N; t, k, v_1 v_2 \ldots v_k)$, is an $N \times k$ array where $v_1 v_2 \ldots v_k$ is a cardinality vector that indicates the values for every column [6]. The MCA has the following properties:

1. Each column i $(1 \le i \le k)$ contains only elements from a set S_i with $|S_i| = v_i$.
2. The rows of each $N \times t$ sub-array cover all t-tuples of values from the t columns at least once.

We use a shorthand notation to describe an MCA, it can be written like MCA $(N; t, k, w_1^{r_1} w_2^{r_2} \ldots w_s^{r_s})$ where $k = \sum_{i=1}^{s} r_i$ and $w_j \in \{v_1, v_2, \ldots, v_k\}$, for all $1 \le j \le k$. The minimum N for which there exists an MCA is called mixed covering array number $\mathrm{MCAN}(k, t, v_1 v_2 \ldots v_k)$.

To illustrate the use of a MCA with a simple instance, suppose that we want to verify a Switch WLAN in four different aspects: monitoring, management, maintenance and safety, the first aspect has three possible values and the rest of them have two possible values as shown in Table 1. Every possible value of each parameter is labeled like 0, 1 or 2 as the case.

Table 1. Parameters of a Switch WLAN, each with three possible values and the rest with two

	Monitoring	**Management**	**Maintenance**	**Safety**
$0 \rightarrow$	PC	Load balancing	Interference	Denial of service
$1 \rightarrow$	Access points	Connection	Barriers	Ad-hoc Networks
$2 \rightarrow$	Sensors			

The MCA(6; 2, 4, $3^1 2^3$) is a test suite that covers all interactions between pairs in the Switch WLAN where every row indicates the configuration of a test case. The mapping of the MCA to the corresponding software test suite is shown in Table 2.

Table 2. Mapping of the MCA(6; 2, 4, $3^1 2^3$) to the corresponding pair-wise test suite

	Monitoring	**Management**	**Maintenance**	**Safety**
0 0 1 1 →	PC	Load balancing	Barriers	Ad-hoc Networks
0 1 0 0 →	PC	Connection	Interference	Denial of service
1 0 0 1 →	Access points	Load balancing	Interference	Ad-hoc Networks
1 1 1 0 →	Access points	Connection	Barriers	Denial of service
2 0 1 0 →	Sensors	Load balancing	Barriers	Denial of service
2 1 0 1 →	Sensors	Connection	Interference	Ad-hoc Networks

3 Related Work

There are exact and approximated approaches for the construction of CAs [11]. The approximated methods do not guarantee that the provided solution is always optimal, in the other hand, the exact methods guarantee that the solution delivered is optimal. A repository of CAs is available on line [28], some of them are optimal or near to the optimal.

Some of the non-exact methods include genetic algorithms (GA), simulated annealing (SA), tabu search (TS), hill climbing (HC), ant colony optimization algorithm (ACO) and great deluge (GD). The greedy methods have been implemented in the algorithms Automatic Efficient Test Generator (AETG), Deterministic Density Algorithm (DDA), Test Case Generator (TCG) and In Parameter Order (IPO) which was subsequently extended to In Parameter Order General (IPOG). Another approaches like algebraic methods, constraint programming (CP) and B&B have also been applied.

Shiba et al. [25] implemented GA, ACO and SA. Stardom [26] implemented SA, TS and GA; from these last approaches SA was the one that reported the best results. Stardom and Bryce [4] emphasize that SA and TS have constructed many of the CA optimal or near to the optimal. Nurmela [22] also used TS for constructing CA and MCA and reported some upper bounds for them, Walker and Colbourn [16] employed TS using permutation vectors. Likewise Bryce and Colbourn [2] implemented an hybrid technique of greedy methods with the metaheuristics TS, HC, SA and GD.

Some greedy algorithms generate one test at a time; some examples are AETG [5], DDA [8], IPO [19], IPOG [18], TCG [30]. The algorithms AETG and TCG have some variants like the ones implemented by Cohen et al. William and Probert [32] proposed a method for constructing CAs based on algebraic methods and combinatorial theory.

With respect to exact approaches, Bracho et. al. [1] implemented a backtracking algorithm called B&B for constructing binary CA of variable strength. Yang and Zang [33] used a retrospective algorithm and incorporated it in a tool called EXACT.

This paper proposes an algorithm based on a TS approach to construct MCAs; the contribution of this new approach is the use of a mixture of neighborhood functions instead of a single one to form new solutions. The next section describes in depth the proposed approach.

4 Proposed Approach

The Tabu Search (TS) meta-heuristic is a local search optimization approach that copes with different problems of combinatorial optimization [24]. The TS was proposed by Glover and Laguna [10]. The general idea behind TS is based on a tabu list. The tabu list keeps track of some of the previous movements done to transform the actual solution s into a new solution s'. Then, every time that a new solution is created the movements found in the tabu list may be avoided. The tabu list can be defined as a *memory* that stores information of forbidden moves. Another distinguishing feature of TS is the *aspiration criteria*, which allows the use of movements in the tabu list when they can create better solutions than the best solution found so far.

The movements that form new solutions comes from the *neighborhood function.* A neighborhood function $f(s)$ is a set of movements that allow the generation of new solutions s' given a current solution s. The solution s' is called a neighbor of s. Whenever some of the movements performed by the neighborhood function are random, the set of neighbors derived from s are called the *neighborhood* and denoted by $\mathcal{N}(s)$. When more than one neighbor are possible, the use of an *objective function* that evaluates their cost will decide the one that will be chosen as the new solution s'. In optimization problems, the best solution will be the one that minimizes or maximizes the value resulting from objective function.

This paper proposes the algorithm Mixed Tabu Search (MiTS) to construct MCAs of variable strength. The MiTS algorithm is a new approach based on TS. The key feature of MiTS is the use of a mixture of neighborhood functions to form $\mathcal{N}(s)$. The elements that defines the MiTS algorithm are: a) the initial solution $\texttt{s}_0$; b) the tabu list; c) the stop criterion; d) the neighborhood function $\texttt{F(s,}\rho_1\texttt{, }\rho_2\texttt{, }\rho_3\texttt{)}$; and e) the evaluation function $\texttt{C(s)}$.

The following paragraphs will describe each of the elements of the algorithm MiTS. The description is done given the matrix representation of an MCA. An MCA can be represented as a matrix $\mathcal{M}$ of size $N \times k$. Each cell $m_{i,j} \in \mathcal{M}$ can take values from $\{0, 1, ..., v_j - 1\}$, where v_j is the cardinality of the alphabet corresponding to the parameter j in the instance.

The initial solution s_o can be constructed by choosing each value $m_{i,j}$ of the matrix $\mathcal{M}$ randomly or by generating $\mathcal{M}$ as a matrix with maximum Hamming distance.

Let r_i be a row of the matrix $\mathcal{M}$ (then, the row is a vector of size k). To generate a random matrix $\mathcal{M}$ of maximum Hamming distance the following steps are performed: a) generate the first row r_1 at random; b) generate two rows c_1, c_2 at random, which will be candidate rows; c) select the candidate row c_i that maximizes the hamming distance (which is defined in Equation 1) and added to the i^{th} row of the matrix $\mathcal{M}$; d) repeat from step b until $\mathcal{M}$ is completed.

$$g(r_i) = \sum_{s=1}^{i-1} \sum_{v=1}^{k} h(m_{s,v}, m_{i,v}), \qquad \textbf{where } h(m_{s,v}, m_{i,v}) = \begin{cases} 1 & \textbf{if } m_{s,v} \neq m_{i,v} \\ 0 & \textbf{otherwise} \end{cases} \tag{1}$$

The hamming distance defined in Equation 1 is the number of different symbols between two rows. An example is shown in Table 3; the number of symbols different between rows r_1 and c_1 are 2 and between r_2 and c_1 are 3 summing up 5. Then, the hamming distance for the candidate row c_1 is 5.

Table 3. Example of the hamming distance between two rows r_1, r_2 that are already in the matrix $\mathcal{M}$ and a candidate row c_1

	$\mathcal{M}$
r_1	0 1 0 1 0 0 1 1
r_2	1 1 0 0 0 1 1 1
c_1	0 1 1 1 0 1 1 1
	$g(c_1) = 5$

The values for tabu list size $\mathcal{T}$ analyzed during the design of the MiTS algorithm are shown in column 1 of Table 4; these values depends on the size of the matrix $\mathcal{M}$ and in $v_{max} = \prod_{i=1}^{i=t} w_i$ (where w_i is the i^{th} cardinality of alphabet in decreasing order). An element that belongs to the tabu list if formed by the tuple (i, j, v, F) i.e., a movement is tabu if the symbol v is generated more than once by the same neighborhood function $F(s)$ in the cell $m_{i,j} \in \mathcal{M}$.

The evaluation function $C(s)$ of a solution s is defined as the number of combination of symbols missing in the matrix $\mathcal{M}$. Then, the expected solution will be zero missings.

The stop criterion for the MiTS algorithm is a given number of evaluations $\mathcal{E}$of the objective function. The values considered are shown in column 2 of Table 4. An alternative stop criterion is when the MCA has been created i.e., when the number of missing is 0.

Table 4. Values for tabu list size $\mathcal{T}$ and number of evaluations $\mathcal{E}$ that were analyzed when designing the MiTS algorithm

$\mathcal{T}$	$\mathcal{E}$
$\mathcal{T}_1 = N * k * v_{max}/8$	$\mathcal{E}_1 = N * k * v_{max}^t * 100$
$\mathcal{T}_2 = N * k * v_{max}/10$	$\mathcal{E}_2 = N * k * v_{max}^t * 150$
$\mathcal{T}_3 = N * k * v_{max}/12$	$\mathcal{E}_3 = N * k * v_{max}^t * 200$

The three neighborhood functions proposed in this paper are $F_1(s), F_2(s)$ and $F_3(s)$. The function $F_1(s)$ randomly chooses a position (i, j) of the matrix $\mathcal{M}$ and carries out all the possible changes of symbol that cell. This functions has $v_j - 1$ possible neighbors. The function $F_2(s)$ selects a column of $\mathcal{M}$ at random makes all possible changes of symbol using only a single cell. The number of neighbors generated with $F_2(s)$ are $(v_j - 1) * N$. Finally, the function $F_3(s)$ makes the change of symbols on each cell of $\mathcal{M}$. The number of different neighbors is $(v_j - 1) * N * k$.

The pseudocode of MiTS is shown in the Algorithm 1. In this algorithm the function `F(s,`ρ_1`,` ρ_2`,` ρ_3`)` makes a roulette-wheel selection with the values ρ_1, ρ_2, ρ_3; the result will indicate which neighborhood function will be used to create a neighbor. The function `NumEvalRequired(s,`ρ_1`,` ρ_2`,` ρ_3`)` will determine the number of evaluations performed by the neighborhood function used by `F(s,`ρ_1`,` ρ_2`,` ρ_3`)` to create a new neighbor.

Algorithm 1. Pseudocode of the MiTS algorithm

```
s ← s_0;
s_best ← s;
while C(s_best) > 0 and e < $\mathcal{E}$ do
    s' ← F(s, $\rho_1$, $\rho_2$, $\rho_3$);
    if C(s') < C(s_best) then
        s_best ← s';
    end
    if NotInTabuList(s') then
        s ← s';
        UpdateTabuList(s, s');
    end
    e ← NumEvalRequired(s, $\rho_1$, $\rho_2$, $\rho_3$);
end
```

The design of the approach presented in this section was based on the premise that using a mixture of neighborhood functions in TS, rather than used just one, improves the construction of MCAs.

5 Experimental Design

MiTS was implemented in C language and compiled with gcc. The instances have been run on a cluster using eight processing nodes, each with two dual-core Opteron Processors. The features of each node are: Processor 2 X Dual-Core AMD, Opteron Processor 2220, 4GB RAM Memory, Operating Systems Red Hat Enterprise Linux 4 64-bit and gcc 3.4 Compiler.

The experiment has the main goal to fine-tune the parameters of the new TS approach such that it achieves a quicker convergence. The parameters that were adjusted are:

– The initial solution s_o.
– The size of tabu list $\mathcal{T}$.
– Maximum number of iterations $\mathcal{E}$.
– Probabilities (ρ_1, ρ_2, ρ_3) for the use of each neighborhood function $F_i(s)$.

In order to find the best combination of values for s_o, $\mathcal{T}$ and $\mathcal{E}$ a MCA with pair interaction was used. The MCA generated 3^2 configurations to be tested. The different configurations are shown in Table 5a.

Table 5. Configurations used for constructing the test cases to fine-tune parameters

(a) Values of s_o, $\mathcal{T}$ and $\mathcal{E}$.

Codification	s_o	$\mathcal{T}$	$\mathcal{E}$
C1	Random way	$\mathcal{T}_1$	$\mathcal{E}_1$
C2	Random way	$\mathcal{T}_2$	$\mathcal{E}_2$
C3	Random way	$\mathcal{T}_2$	$\mathcal{E}_3$
C4	Random way	$\mathcal{T}_3$	$\mathcal{E}_1$
C5	Random way	$\mathcal{T}_3$	$\mathcal{E}_2$
C6	Hamming distance	$\mathcal{T}_1$	$\mathcal{E}_2$
C7	Hamming distance	$\mathcal{T}_1$	$\mathcal{E}_3$
C8	Hamming distance	$\mathcal{T}_2$	$\mathcal{E}_1$
C9	Hamming distance	$\mathcal{T}_3$	$\mathcal{E}_3$

(b) Probabilities for use each neighborhood function.

Codification	ρ_1	ρ_2	ρ_3	**Codification**	ρ_1	ρ_2	ρ_3
C1-P	0.0	0.0	1.0	C12-P	0.4	0.0	0.6
C2-P	0.0	0.2	0.8	C13-P	0.4	0.2	0.4
C3-P	0.0	0.4	0.6	C14-P	0.4	0.4	0.2
C4-P	0.0	0.6	0.4	C15-P	0.4	0.6	0.0
C5-P	0.0	0.8	0.2	C16-P	0.6	0.0	0.4
C6-P	0.0	1.0	0.0	C17-P	0.6	0.2	0.2
C7-P	0.2	0.0	0.8	C18-P	0.6	0.4	0.0
C8-P	0.2	0.2	0.6	C19-P	0.8	0.0	0.2
C9-P	0.2	0.4	0.4	C20-P	0.8	0.2	0.0
C10-P	0.2	0.6	0.2	C21-P	1.0	0.0	0.0
C11-P	0.2	0.8	0.0				

The probabilities ρ_1, ρ_2, ρ_3 for the neighbor functions $F_1(s), F_2(s), F_3(s)$ were obtained by solving the equation 2. The discrete values considered for each probability ρ_i were $\{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ (this represent a granularity of 0.2). The number of different valid configurations of probabilities resulting from solving the equation 2 with a granularity of 0.2 were 21 (they are shown in Table 5b).

$$\rho_1 + \rho_2 + \rho_3 = 1 . \tag{2}$$

A design involving the configurations shown in Tables 5a and 5b was generated; the resulting test set now can be used to test the different parameters of MiTS. The whole set of test cases included a total of $9 * 21 = 189$ different configurations; each configuration was used to construct the instance $MCA(30; 2, 19, 6^1 5^1 4^6 3^8 2^3)$ 31 times. The following section presents some statistical results about the fine-tuning of MiTS using the test set described in this section.

6 Computational Results

The main purpose of this section is to do an experimental comparison of MiTS against the state-of-the-art algorithms for constructing MCAs. In order to do the comparison first, we show the results from the fine-tuning of the main components of MiTS. After that we present the best configuration values for the components; those values will be used by MiTS to construct MCAs taken from a benchmark found in the literature.

During the fine-tuning the instance $MCA(30; 2, 19, 6^1 5^1 4^6 3^8 2^3)$ was solved by MiTS. Each of the 189 valid configurations were tested 31 times with MiTS to construct the desired MCA. The results from this experiment are summarized in Table 6. The columns denote the different configuration values for the components s_0, $\mathcal{T}$ and $\mathcal{E}$ shown in Table 5b. The rows represent the valid probability configurations for the neighborhood functions, shown in Table 5a. Each cell of Table 6 represents one of 189 configurations to be tested to measure the performance of MiTS. The information in each cell contains how many times (expressed as a percentage of the 31 times) MiTS constructed the MCA using that configuration. The results show that, there are a lot of configurations where MiTS could construct the MCA in less than 20% of the times; observe that in this situation we found all the configurations involving only one neighborhood function (like in rows C1-P, C6-P and C21-P). In the other hand, some configurations that involve a mixture of neighborhood functions made MiTS to have the best performance, by constructing the MCA in more than 80% of the times e.g., the configuration (C9, C2-P) and (C9, C3-P).

Table 6. Solved percentage of each configuration based on 31 runs

	C1	C2	C3	C4	C5	C6	C7	C8	C9	**Average**
C1-P	6.45	16.13	9.68	3.23	6.45	3.23	12.90	3.23	6.45	7.53
C2-P	29.03	54.84	54.84	64.52	38.71	64.52	25.81	64.52	90.32	54.12
C3-P	38.71	58.06	74.19	87.10	61.29	67.74	58.06	74.19	87.10	67.38
C4-P	32.26	35.48	45.16	61.29	54.84	54.84	51.61	83.87	67.74	54.12
C5-P	19.35	12.90	45.16	58.06	48.39	45.16	45.16	58.06	83.87	46.24
C6-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C7-P	3.23	9.68	29.03	25.81	19.35	16.13	6.45	41.94	41.94	21.51
C8-P	12.90	38.71	38.71	64.52	41.94	32.26	29.03	48.39	64.52	41.22
C9-P	12.90	32.26	48.39	61.29	45.16	58.06	41.94	64.52	67.74	48.03
C10-P	19.35	6.45	25.81	29.03	25.81	16.13	25.81	41.94	45.16	26.16
C11-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C12-P	3.23	0.00	6.45	3.23	6.45	3.23	0.00	12.90	0.00	3.94
C13-P	0.00	3.23	0.00	9.68	3.23	6.45	3.23	6.45	19.35	5.73
C14-P	0.00	0.00	0.00	0.00	0.00	3.23	0.00	3.23	3.23	1.08
C15-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C16-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C17-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C18-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C19-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C20-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C21-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Average	8.45	12.75	17.97	22.27	16.74	17.67	14.29	23.96	27.50	17.96

According with the results shown in Table 6 the best configuration for MiTS was (C9, C2-P); this configuration could find the solution in 90.32% of the 31 times that it was tried. The values suggested by the configuration (C9, C2-P) are the Hamming distance as s_0, the tabu list size $\mathcal{T}_3$, the number of evaluations $\mathcal{E}_3$ and the values $\{\rho_1 = 0.0, \rho_2 = 0.2, \rho_3 = 0.8\}$ as the probabilities values for the neighborhood functions.

We now proceed to compare the MiTS algorithm with the state-of-the-art approaches for constructing MCAs. The algorithm MiTS was configured with (C9, C2-P). The benchmark was taken from the literature and is shown in the column 1 of Table 7. From column 2 to 9 are presented the results reported by some of the state-of-the-art approaches. The results of constructing the MCAs for the benchmark (using the approach presented in this paper) are shown in column 10. Column 11 shows the best reported MCA sizes and column 12 depicts the best time (in seconds) spent by MiTS. The results show that MiTS found the best reported results in 8 of the 10 instances; in 4 of the 8 instances MiTS improved the best reported results so far.

Table 7. Results of MiTS compared with some results previously reported. The * means that the solution is optimal.

Instance	**TConfig** [31] [2]	**IPO** [27] [2]	**AETG** [5]	**TCG** [5]	**SA** [6]	**GA** [25]	**ACO** [25]	**DDA** [8]	**MiTS**	**Best reported**	**Time [sec.]**
MCA$(N; 2, 5^1 3^8 2^2)$	21	21	19	18	15	15	16	21	15	15*	0.03
MCA$(N; 2, 7^1 6^1 5^1 4^5 3^8 2^3)$	91	48	45	42	42	42	42	43	42	42*	0.26
MCA$(N; 2, 4^{15} 3^{17} 2^{29})$	39	34	41	35	30	37	37	35	30	29	25.22
MCA$(N; 2, 4^1 3^{39} 2^{35})$	29	26	28	26	21	27	27	27	22	21	5.81
MCA$(N; 2, 10^1 9^1 8^1 7^1 \ldots$ $\ldots 6^1 5^1 4^1 3^1 2^1 1^1)$								91	90	90*	0.55
MCA$(N; 2, 8^2 7^2 6^2 5^2)$								70	64	64*	1.87
MCA$(N; 2, 6^6 5^5 3^4)$								56	50	50	3.94
MCA$(N; 2, 4^5 3^4)$								23	19	19	0.13
MCA$(N; 3, 5^2 4^2 3^2)$			114		100	108	106		100	100*	3.21
MCA$(N; 3, 10^1 6^2 4^3 3^1)$			377		360	360	361		360	360*	37.18

7 Conclusions

This paper focused on constructing MCAs with a new approach of TS called MiTS (Mixed Tabu Search), which uses a mixture of neighborhood functions. A fine-tuning of the parameters of MiTS was done, in order to achieve a competitive performance for it. The fine-tuning was carried out for the main components of MiTS which are: initial solution s_0, tabu list size $\mathcal{T}$, maximum number of evaluations $\mathcal{E}$ and the probabilities of selection $\{\rho_1, \rho_2, \rho_3\}$ for the neighborhood functions $F_1(s), F_2(s), F_3(s)$. For each component different values were taken into account.

The best configuration that resulted from the fine-tuning was used in a second experiment to test the performance of MiTS against 8 state-of-the-art algorithms that constructs MCAs. In the second experiment, the performance of MiTS over 10 instances taken from the literature was measured. The results showed that MiTS results are of size equal to the best reported values in 8 of the 10 instances; in 4 of these cases MiTS improved the best solution known. In general, MiTS showed to be competitive in comparison with the state-of-the-art algorithms reported in the literature when considered the time spent in the construction of MCAs and the quality of the MCAs (in terms of its size).

In conclusion, the use of an appropriate mixture of neighborhood functions with TS, combined with an adequate fine tuning of the parameters for its main components, can lead to the design of TS implementation with a better performance.

Acknowledgments. This research was partially funded by the following projects: CONACyT 58554- Cálculo de Covering Arrays, 51623-Fondo Mixto CONACyT y Gobierno del Estado.

References

1. Bracho-Rios, J., Torres-Jimenez, J., Rodriguez-Tello, E.: A new backtracking algorithm for constructing binary covering arrays of variable strength. In: Hernández Aguirre, A., et al. (eds.) MICAI 2009. LNCS (LNAI), vol. 5845, pp. 397–407. Springer, Heidelberg (2009)
2. Bryce, R.C., Colbourn, C.J.: The density algorithm for pairwise interaction testing: Research articles. Softw. Test. Verif. Reliab. 17(3), 159–182 (2007)
3. Bryce, R.C., Colbourn, C.J.: One-test-at-a-time heuristic search for interaction test suites. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 1082–1089. ACM, New York (2007)
4. Bryce, R.C., Colbourn, C.J., Cohen, M.B.: A framework of greedy methods for constructing interaction test suites. In: Inverardi, P., Jazayeri, M. (eds.) ICSE 2005. LNCS, vol. 4309, pp. 146–155. Springer, Heidelberg (2006)
5. Cohen, D.M., Fredman, M.L., Patton, G.C.: The aetg system: An approach to testing based on combinatorial design. IEEE Transactions on Software Engineering 23, 437–444 (1997)
6. Cohen, M.B., Gibbons, P.B., Mugridge, W.B., Colbourn, C.J.: Constructing test suites for interaction testing. In: Proc. of the 25th Int. Conf. on Software Engeneering (ICSE 2003), pp. 38–48 (2003)
7. Colbourn, C.J.: http://www.public.asu.edu/~ccolbou/src/tabby/6-4-ca.html
8. Colbourn, C.J., Cohen, M.B., Turban, R.C.: A deterministic density algorithm for pairwise interaction coverage. In: Proc. of the IASTED Intl. Conference on Software Engineering (2004)
9. Fisher, R.A.: The arrangement of field experiments. Journal of Ministry of Agriculture of Great Britain 33, 503–513 (1926), http://www.library.adelaide.edu.au/digitised/fisher/48.pdf
10. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Dordrecht (1998)
11. Grindal, M., Offutt, J., Andler, S.F.: Combination testing strategies: a survey. Software testing verification & reliability 15, 167–199 (2005)
12. Hartman, A., Ben, I.: Graph Theory, combinatorics and algorithms interdisciplinary applications, 4th edn. Springer, US (2005)
13. Hartman, A., Raskin, L.: Problems and algorithms for covering arrays. Discrete Mathematics 284, 149–156 (2004)
14. Hedayat, A.S., Sloane, N.J.A., Stufken, J.: Orthogonal Arrays: Theory and Applications. Springer, New York (1999)
15. Hnich, B., Prestwich, S.D., Selensky, E., Smith, B.M.: Constraint models for the covering test problem. Constraints 11(2-3), 199–219 (2006)
16. Walker II, R.A., Colbourn, C.J.: Tabu search for covering arrays using permutation vectors. Journal of Statistical Planning and Inference 139(1), 69–80 (2009); Special Issue on Metaheuristics, Combinatorial Optimization and Design of Experiments
17. Kuhn, D.R., Wallance, D.R., Gallo Jr., A.M.: Software fault interactions and implications for software testing. IEEE Transactions on Software Engineering 30, 418–421 (2004)

18. Lei, Y., Kacker, R., Kuhn, R.D., Okun, V., Lawrence, J.: Ipog: A general strategy for t-way software testing. In: 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS 2007), pp. 549–556 (2007)
19. Lei, Y., Tai, K.C.: In-parameter-order: A test generation strategy for pairwise testing. In: Proceedings. Third IEEE International High-Assurance Systems Engineering Symposium, pp. 254–261 (1998)
20. Levenson, N.G., Turner, C.S.: An investigation of the therac-25 accidents. IEEE Computer 26, 18–41 (1993)
21. Lopez-Escogido, D., Torres-Jimenez, J., Rodriguez-Tello, E., Rangel-Valdez, N.: Strength two covering arrays construction using a sat representation. In: Gelbukh, A., Morales, E.F. (eds.) MICAI 2008. LNCS (LNAI), vol. 5317, pp. 44–53. Springer, Heidelberg (2008)
22. Nurmela, K.J.: Upper bounds for covering arrays by tabu search. Discrete applied mathematics 138, 143–152 (2004)
23. National Institute of Standards and Technology. The economic impacts of inadecuate infraestructure for software testing. U.S. Deparment of Commerce (May 2002)
24. Rodrigues, L.C.A., Weller, T.R.: Cell formation with alternative routings and capacity considerations: A hybrid tabu search approach. In: Gelbukh, A., Morales, E.F. (eds.) MICAI 2008. LNCS (LNAI), vol. 5317, pp. 482–491. Springer, Heidelberg (2008)
25. Shiba, T., Kikuno, T., Tsuchiya, T.: Using artificial life techniques to generate test cases for combinatorial testing. In: Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC 2004), vol. 1, pp. 28–30 (2004)
26. Stardom, J.: Metaheuristics and the Search for Covering and Packing Arrays. PhD thesis, Simon Fraser University (2001)
27. Tai, K.C., Lei, Y.: A test generation strategy for pairwise testing. IEEE Transactions on Software Engineering 28, 109–111 (2002)
28. Torres-Jimenez, J.: http://www.tamps.cinvestav.mx/~jtj/authentication.php
29. Torres-Jimenez, J., de Alfonso, C., Hernández, V.: Computation of ternary covering arrays using a grid. In: AACC, pp. 240–246 (2004)
30. Tung, Y.W., Aldiwan, W.S.: Automating test case generation for the new generation mission software system. In: Proceedings Aerospace Conference IEEE, vol. 1, pp. 431–437 (2000)
31. Williams, A.W.: Determination of test configurations for pair-wise interaction coverage. In: TestCom 2000: Proceedings of the IFIP TC6/WG6.1 13th International Conference on Testing Communicating Systems, Deventer, The Netherlands, pp. 59–74. Kluwer, B.V., Dordrecht (2000)
32. Williams, A.W., Probert, R.L.: A practical strategy for testing pair-wise coverage of network interfaces. In: Seventh Intl. Symp. on Software Reliability Engineering, pp. 246–254 (1996)
33. Yan, J., Zhang, J.: A backtraking search tool for constructing combinatorial test suites. The Journal of Systems and Software (2008), doi:10.1016/j.jss.2008.02.034

On the Best Evolutionary Wavelet Based Filter to Compress a Specific Signal

Oscar Herrera Alcántara

Departamento de Sistemas, Universidad Autónoma Metropolitana Azcapotzalco
Av. San Pablo 180, Col. Reynosa Tamaulipas, 02200, México, D.F.
oha@correo.azc.uam.mx
http://ia.azc.uam.mx

Abstract. This work presents an application of a genetic algorithm in the design of digital filters used to implement the discrete wavelet transform. The best compression of a transformed signal is achieved when its power is described by the smallest number of transformation coefficients. The individuals of the genetic algorithm aim to reach this target, at the same time that they reduce the error of the reconstructed signal. In the experiments we worked with grayscale images, and we compared the performance of evolutionary and Daubechies filters. Experimental results show the feasibility and convenience of finding custom wavelets for each image, and support the idea that there is a suitable wavelet to compress any given signal.

Keywords: Wavelet Transform, Digital Filtering, Data Compression, Genetic Algorithm.

1 Introduction

The continuous wavelet transform allows to decompose a function/signal $f(t)$ in a set of basis $\Psi^*_{s,t}(t)$[1] called wavelets. Equation (1) shows how a function $f(t)$ can be decomposed into a set of basis function wavelets:

$$\gamma(s,t) = \int f(t)\Psi^*_{s,t}(t)dt \tag{1}$$

The wavelets comes from a mother wavelet by scaling and translation, that is:

$$\Psi(s,t) = \frac{1}{\sqrt{s}}\Psi(\frac{t-\tau}{s}) \tag{2}$$

where s is a scale factor, and τ is the translation factor. The factor $\frac{1}{\sqrt{s}}$ gives energy normalization across the different scales. An important difference with Fourier analysis is that the wavelet basis are not specific, but the analysis describes the general properties of the wavelets. Consider the expansion of equation

[1] Denotes complex conjugation.

(1) into the Taylor series at $t = 0$ until order n for the continuous wavelet transform, it takes us to equation (3):

$$\gamma(s,0) = \frac{1}{\sqrt{s}}[\sum_{p=0}^{n} f^p(0) \int \frac{t^p}{p!}\Psi(\frac{t}{s})dt + O(n+1)] \quad (3)$$

where $f^{(p)}$ denotes the $p-th$ derivative of f, and the term $O(n+1)$ means the rest of the expansion. Let $M_p = \int t^p \Psi(t)dt$, it is possible to demostate that $\gamma(s,0)$ can be reexpressed as:

$$\gamma(s,0) = \frac{1}{\sqrt{s}}[f(0)M_0(s) + \frac{f^(1)(0)M_1 s^2}{1!} + \ldots + \frac{f^n(0)M_n s^{n+1}}{n!} + O(s^{n+2})] \quad (4)$$

A property of wavelets is that $M_0 = 0$, so the first term in the equation (4) is zero, and for a given wavelet the other moments migth be (near to) zero as well, then the wavelet transform coefficients $\Psi(s,\tau)$ will decay as fast as s^{n+2} for a smooth signal $f(t)$. If the M_N moments are lower than a threshold value, they are considered negligible and the decomposition of a signal concentrates its power in a small number of coeficients. This property is desirable from the point of view of compression because that means that we were able to find an appropriate vector space for the representation of the analyzed function but, what is the best wavelet based filter for compressing a given signal? Certainly, this work is not the first in dealing with this idea (see for example [1], [2]). In fact, motivated by related works, we propose to choose parametric evolutionary wavelets for specific discrete signals focused on achieving maximum compression. Typically compressors aim to be as fast as possible and they must leave the compression as a second priority, because there is a compromise between achieving greater compression and the time invested in it [3]. Symmetric systems require that the compression time be approximately equal to the decompression time, but systems like the Web, digital encyclopedias, musical CD's and other, which concentrate a large amount of multimedia, can spend larger time and computing resources in the compression than in the decompression task, and the latter will be done by many users with limited resources systems.

The article is organized in sections: Section 2 describes basic concepts about wavelets, the discrete wavelet transform, and its relation with bank filters. In subsection 2.4 we discuss the criteria to choose a parametric wavelet that achieves the best modelling of two-dimensional signals (grayscale images) considering the energy conservation. In section 3 we explain some experiments with grayscale images and compare the performance of the evolutionary and Daubechies-4 wavelet based filters. In section 4 we report the results of the experiments, and finally, in section 5 we comment about some conclusions and future work.

2 Wavelets, Filter Banks and Multiresolution Signal Processing

Wavelets are mathematical functions associated with scaling functions that have the property of being described as linear combinations of traslated and dilated versions of themselves. The most basic wavelet is the Haar wavelet which is associated with the Haar scaling function. The Haar wavelet function is described by:

$$\psi(t) = \begin{cases} 1, \ if \ 0 \leq t < \frac{1}{2} \\ -1, if \ \frac{1}{2} \leq t < 1 \\ 0, \quad otherwise \end{cases} \tag{5}$$

and the Haar scaling function is described by:

$$\psi(t) = \begin{cases} 1, if \ \ 0 \leq t < 1 \\ 0, \quad otherwise \end{cases} \tag{6}$$

It can be shown that the Haar scaling function $\varphi(t)$ satisfies the condition:

$$\varphi(t) = \varphi(2t) + \varphi(2t - 1) \tag{7}$$

that allows to analyze a signal at different levels of decomposition because we can express $\varphi(t)$ using $\varphi(2t)$, $\varphi(2t)$ using $\varphi(4t)$, and so on.

2.1 Discrete Wavelet Transform

The continuous wavelet transform described in equation (1) has several drawbacks to make it practical, one of them is that for most functions the wavelet transforms have no analytical solutions. A second is that when it is calculated produces high redundance because of the continuously shifting between the scalable function and the signal, and altought this may be useful in some applications, it is not for data compression. To overcome this problem discrete wavelets are introduced. Discrete wavelets are not continuously scalable and translatable, but can only be scaled and translated in discrete steps. The modification of equation (1) gives [4] :

$$\psi_{j,k}(t) = \frac{1}{\sqrt{s_0^j}} \psi(\frac{t - k\tau_0 s_0^j}{s_0^j}) \tag{8}$$

In equation (8), j and k are integers and $s_0 > 1$ is a fixed dilation step. The translation factor τ_0 depends on the dilation step. The effect of discretizing the wavelet is that the time-scale space is now sampled at discrete intervals. When $s_0 = 2$ the sampling of the frequency axis corresponds to dyadic sampling, and for the translation factor $\tau_0 = 1$ we have dyadic sampling of the time axis. The discrete wavelets can be made orthogonal to their own dilations and translations by special choices of the mother wavelet, which means:

$$\int \psi_{j,k}(t) \psi_{m,n}{}^*(t) dt = 1, \quad if \ j = m \ and \ k = n, \ but \ 0 \ otherwise \tag{9}$$

A square integrable signal can be reconstructed by summing the orthogonal wavelet basis functions, weighted by the wavelet transform coefficients [5]:

$$f(t) = \sum_{j,k} \gamma(j,k) \psi_(j,k)(t) \tag{10}$$

this is the inverse wavelet transform for discrete wavelets.

2.2 Multiresolution Analysis and Filters

The condition described in equation (7) for the Haar wavelet is a particular case of the so-called multiresolution formulation [6] or two-scale relation [5]:

$$\varphi(t) = \sum_k h_k \varphi(2t - k) \tag{11}$$

where h_k is a real sequence with even length. For example, in the Haar scaling function $\{h_k\} = \{1, 1\}$. With the two-scale relation the scaling function at a certain scale can be expressed in terms of translated scaling functions at the next smaller scale, that is:

$$\varphi(2^j t) = \sum_k h_{j+1}(k) \varphi(2^{j+1} t - k) \tag{12}$$

And when the first scaling function is replaced by a set of wavelets, we can also express the wavelets at level j in terms of translated scaling functions at the next scale $j + 1$:

$$\psi(2^j t) = \sum_k g_{j+1}(k) \varphi(2^{j+1} t - k) \tag{13}$$

Now a square integrable function $f(t)$ can be expressed in terms of dilated and translated scaling functions at a scale j as:

$$f(t) = \sum_k \lambda_j(k) \phi(2^j t - k) \tag{14}$$

If in equation (14) we step up a scale to $j - 1$, we have to add wavelets in order to keep the same level of detail then the signal $f(t)$ can be expressed as:

$$f(t) = \sum_k \lambda_{j-1}(k) \phi(2^{j-1} t - k) + \sum_k \gamma_{j-1}(k) \psi(2^{j-1} t - k) \tag{15}$$

As the scaling function $\phi_{j,k}(t)$ and the wavelets $\psi_{j,k}(t)$ are chosen to be orthonormal, then the coefficients $\lambda_{j-1}(k)$ and $\gamma_{j-1}(k)$ can be calculated with the next inner products:

$$\lambda_{j-1}(k) = \langle f(t), \phi_{j,k} \rangle \tag{16}$$

$$\gamma_{j-1}(k) = \langle f(t), \psi_{j,k} \rangle \tag{17}$$

If we replace $\phi_{j,k}(t)$ and $\psi_{j,k}(t)$ in (16) and (17), we get the next expressions

$$\gamma_{j-1}(k) = \sum_m h(m - 2k) \lambda_j(m) \tag{18}$$

$$\gamma_{j-1}(k) = \sum_m g(m - 2k) \gamma_j(m) \tag{19}$$

Equations (18) and (19) enclose an important result [6], they mean that the wavelet and scaling function coefficients on a certain scale can be found by calculating a weighted sum of the scaling function coefficients from the previous scale. In practice, the original sampled signal $f(k)$ is equal to $\lambda(k)$ at the largest scale. The sequence $\{h(k)\}$ satisfying equation (12) is called the scaling filter associated to $\phi(x)$, whereas the sequence $\{g(k)\}$ satisfying equation (13) is called the wavelet filter associated to $\phi(x)$, and the next condition is accomplished:

$$g_k = (-1)^k h_{1-k} \tag{20}$$

In the frequency domain equation (12) and (13) become

$$\Phi(\omega) = H(\frac{\omega}{2})\Phi(\frac{\omega}{2}) \tag{21}$$

$$\Psi(\omega) = G(\frac{\omega}{2})\Phi(\frac{\omega}{2}) \tag{22}$$

where $H(\omega)$ and $G(\omega)$ are the Fourier transform of h_k and g_k, respectively and they are 2π periodic. For orthonormal multiresolution analysis, equation (12) and equation (13) represent the impulse responses of quadrature mirror filters (QMF) that have the following properties:

$$|H(\omega)|^2 + |G(\omega)|^2 = 1 \tag{23}$$

and

$$H(\omega)H^*(\omega + \pi) + G(\omega)G^*(\omega + \pi) = 0 \tag{24}$$

In the frequence domain, equation (20) means:

$$G(\omega) = e^{i\omega}H^*(\omega + \pi) \tag{25}$$

With this conditions, the sequences $\{h(k)\}$ and $\{g(k)\}$ forms a bank filter where the first acts as a low-pass step, and the second as a high-pass step. Therefore, the bank filter provides a coarse and detailed decomposition of a signal at different resolution levels. The 2-step size in the variable k in equations (18) and (19) is called the subsampling property, what means that after a scaling filter follows a downsampling process. Now, the discrete wavelet transform can be calculated by an iterative filtering and downsampling process, as shown in the left part of Figure 1. The reconstruction of the original sampled signal is given by upsampling-digital filtering steps as shown in the rigth part of Figure 1.

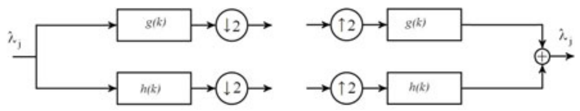


Fig. 1. Filter bank for the discrete wavelet transform: forward step (left) and reverse step (right)

2.3 Filter Sequences

It has been shown that the sequence $\{h(k)\}$ must satisfy the following conditions:

$$\sum h_k = 2 \tag{26}$$

$$\sum h_k h_{k+2m} = 2\delta_m \tag{27}$$

$$\sum (-1)^k k^m h_k = 0, for\ m = 0, 1, \ldots, M-1 \tag{28}$$

where $M \geq 1$ and δ_m is the discrete Kronecker delta function. M is equal to the number of vanishing moments of the wavelet corresponding to $\phi(x)$ and, as well as in the continuous wavelet transform, it controls the compactess of the wavelet representation of any given signal. $\sqrt{2^j}\psi(2^j x - k)_{k,j\in Z}$ forms an orthonormal basis for $L^2(R)$, the set of square integrable functions, and states the relation between the multiresolution analysis and the wavelet analysis. For a filter of length four the h_k coefficients depend of the single parameter[7] $\theta_1 \in [0, 2\pi]$ as described in the next:

$$h_0 = g_3 = sin\theta_1 sin(\theta_1 - \frac{\pi}{4}) \tag{29}$$

$$h_1 = -g_2 = sin\theta_1 sin(\theta_1 + \frac{\pi}{4}) \tag{30}$$

$$h_2 = g_1 = cos\theta_1 sin(\theta_1 + \frac{\pi}{4}) \tag{31}$$

$$h_3 = -g_0 = cos\theta_1 sin(\theta_1 - \frac{\pi}{4}) \tag{32}$$

The filters obtained with equations (29) to (32) allow perfect reconstruction, regardless of the θ parameter value. If $\theta_1 = \frac{\pi}{12}$ the h_k coefficients correspond to those given by the Daubechies-4 filter[8]. The Duabechies filter design criterion focuses on the maximization of the vanishing moments, and the smoothness of the wavelets which is measured with the number of derivatives they have. For example, the simplest Haar (Daubechies-2) wavelet is not continuous and therefore non-diferenciable, and other is the Daubechies-6 wavelet that has a second order derivative. Larger Daubechies filters/wavelets have higher order derivatives. In our approach, we propose to use a genetic algorithm to optimize the parameters in order to achieve maximum compression. Our wavelets are called *evolets* (from evolutionary wavelets), and the main idea is to identify the optimal free parameters such that the corresponding filters give place to vector spaces that model in the best way a given signal (and consequently there will be a greater number of negligible transformation coefficients and compression). Larger filters have more free parameters, and in fact a compactly supported orthonormal wavelet of support size equal to $2N - 1$ is parametrized by $N - 1$ free parameters[9]. For example, the Haar wavelet has one vanishing moment and no free parameter, the Daubechies-4 filter ($N = 2$) has a single free parameter as is described in equations (29) to (32), and the Daubechies-6 filter ($N = 3$) has two free parameters.

2.4 Image Processing and Energy Conservation

An image is a two-dimensional function $f(x, y)$ with discrete values $x \in [0, w]$ and $y \in [0, h]$, where w is the horizontal dimension or width, and h is the vertical dimension or heigth. Each element at position (x, y) is called pixel. In case of grayscale images $f(x, y)$ takes values in $[0, 255]$, where a zero value corresponds to a black pixel, whereas the maximum value 255 is a white pixel. The two-dimensional discrete wavelet transform is calculated applying a horizontal transformation step followed from a vertical transformation step. This gives us a pyramidal effect, that gives places to four subbands HH, HL, LH and LL, with $\frac{w}{2}\text{x}\frac{h}{2}$ wavelet coefficients. The HH subband stores the most detailed information of the image, and the LL subband stores the most coarse information. A second transformation level is applied to the LL subband, that generates other four subbands, and the process continues while the width and heigth of the new subbands are larger than the filter length. The two-dimensional inverse wavelet transform is calculated in reverse order, starting at the top level of the pyramidal subbands. The reconstruction introduces floating point operations, precision errors and filtering errors that can be measured with the RMS error comparing the original and the reconstructed images. Given an one-dimensional signal with samples $f_0, f_1, \ldots, f_n$, its energy is defined as

$$E = \sum_{i=1}^{n} f_i^2 \tag{33}$$

The cumulative energy of the signal is given by

$$E_c = \sum_{i=1}^{r} f_i^2, \qquad for \quad 0 \leq r \leq n \tag{34}$$

When the wavelet tranform is aplied, the scaling factor can be chosen to conserve the energy value at each level (and of course along all the levels) and therefore it can be used to measure the efficiency of energy concentration. The higher the concentration of energy in the lowest number of coefficients, the higher the quality of the filter and the number of negligible coefficients, and consequently it is more useful to compress a given signal[10]. We can sort the transformation coefficients from largest to smallest and measure the cumulative energy. When it reaches an aceptable value of the total, the remaining coefficients can be ignored, and this determines a threshold for the coefficients that will be forced to be zero. A peculiarity of the transformation coefficients, when a good filter is used, is that a small number of them concentrates the most of the energy, therefore one way to measure the efficiency of a filter is counting the number of coefficients which lie below the threshold value, and that do not change substantially the cumulative energy.

3 Experiments

In this work we use grayscale images which are typical in image processing. They have dimensions of 64x64 pixels, that gives us a total of 4096 wavelet transform coefficients. We aims to discover the best h_k sequences calculating the free parameter (θ angle) given by the corresponding parametric equations. For this purpose we appeal to a non-traditional genetic algorithm, the Vasconcelos Genetic algorithm (VGA)[11], although some other can be used. The VGA is characterized by:

- A population of even size P, with a temporal population of size $2P$.
- Pairwise for crossover between the i-th and the $(P-1-i)$-th sorted individuals, $i \in [0, \frac{P}{2}-1]$.
- Deterministic selection, the best P individuals from the temporal population.
- Anular crossover.

Each individual models a value for the parameter θ in the interval $[0, 2\pi]$ using 32 bits of precision. The fitness function measures the number of non-zero coefficients whose value is higher than the threshold. The threshold value is automatically calculated once the transformed coefficients are sorted in descending order of magnitude, and the cumulative energy divided by the total energy is higher than 0.999999. The parameters for the GA execution were: Number of generations $G = 410$, Probability of mutation $Pm = 0.01$, Probability of crossover $Pc = 0.95$, and Population size $P = 8$ individuals. The evolutionary optimization process to compress an image requires $2PG$ two-dimensional wavelet transformations. Consider that the wavelet transform for a vector with n components has $O(n)$ complexity, and that we do not require to apply the inverse wavelet transform because the filters provide perfect reconstruction. In the experiments, the execution time required for the VGA to calculate each parameter is near to one minute running on a Debian/Linux PC, with 2 GB RAM and Intel Celeron 2.2 GHz processor.

4 Results

In Table 1 we compare the number of negligible coefficients (3^{th} column) between the evolutionary filters and the Daubechies-4 filter. The RMSE for the evolutionary wavelets (evolets) and Daubechies-4 are practically zero in all cases with a 10 digits precision.[2] As we can appreciate, in all the cases, the number of negligible coefficients for the evolutionary filters is higher than those of Daubechies-4. That means that evolutionary filters concentrate the energy and model the requiered vector space in a better way than those generated by Daubechies-4 filters. The scaling and the wavelet functions for the filters of Table 2 can be obtained with the iterative cascade algorithm [12][13]. They are shown in Figures 2, 3, and 4.

Note that an image with horizontal or vertical lines or squares, such as the frymire, gives a wavelet similar to the Haar filter. Experimentally we work with

[2] The filters provides perfect reconstruction but in the practice, the reconstruction error is given by the precision used in the implementations. Experimentally we note that at least 8 decimal precision digits are required to get RMSE $< 10^{-10}$, for the Daubechies and evolets filter sequences.

Table 1. Performance of the Evolutionary and Daubechies-4 filters with energy concentration on grayscale images

Image	Filter	#Coef. $<$ Th (max. 4096)	Ep/Ef	Th
1. Babbon.pgm	Evolet	244	0.999999	0.7778971996
	Daubechies	220	0.999999	0.7990381057
2. Barbara.pgm	Evolet	361	0.999999	0.705232105
	Daubechies	348	0.999999	0.713511222
3. Clegg.pgm	Evolet	289	0.999999	1.0653692000
	Daubechies	251	0.999999	1.1312476869
4. Frymire.pgm	Evolet	365	0.999999	0.9938901303
	Daubechies	279	0.999999	1.0866968793
5. Lena.pgm	Evolet	469	0.999999	0.6745043947
	Daubechies	464	0.999999	0.6662658774
6. Monarch.pgm	Evolet	580	0.999999	0.5139496478
	Daubechies	567	0.999999	0.5122595264
7. Peppers.pgm	Evolet	404	0.999999	0.7384878482
	Daubechies	392	0.999999	0.7030444566
8. sail.pgm	Evolet	256	0.999999	0.8169891336
	Daubechies	240	0.999999	0.7855762114

Table 2. Free parameter and the corresponding evolutionary filter sequences h_k

Image	Free parameter	h_0	h_1	h_2	h_3
1.	1.22321408	0.52386620	0.82365289	0.18324057	-0.11654611
2.	1.03044130	0.61077362	0.78231644	0.09633315	-0.07520965
3.	3.97359197	0.01685386	-0.01608620	0.69025291	0.72319298
4.	3.92145800	-0.00195072	0.00196150	0.70905750	0.70514523
5.	1.29121813	0.49152851	0.83413935	0.21557826	-0.12703257
6.	1.28124697	0.49631357	0.83273971	0.21079320	-0.12563293
7.	1.35440831	0.46090502	0.84189305	0.24620175	-0.13478627
8.	1.35527121	0.46048359	0.84198550	0.24662318	-0.13487872

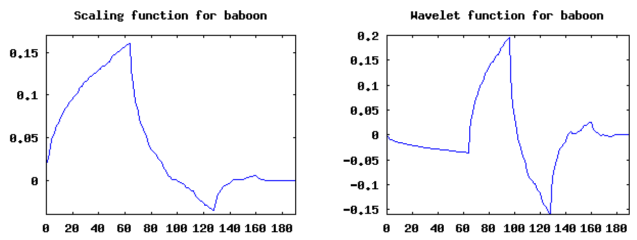


Fig. 2. Pairs of scaling (left) and wavelet (right) functions for the filters of Table 2. (baboon image).

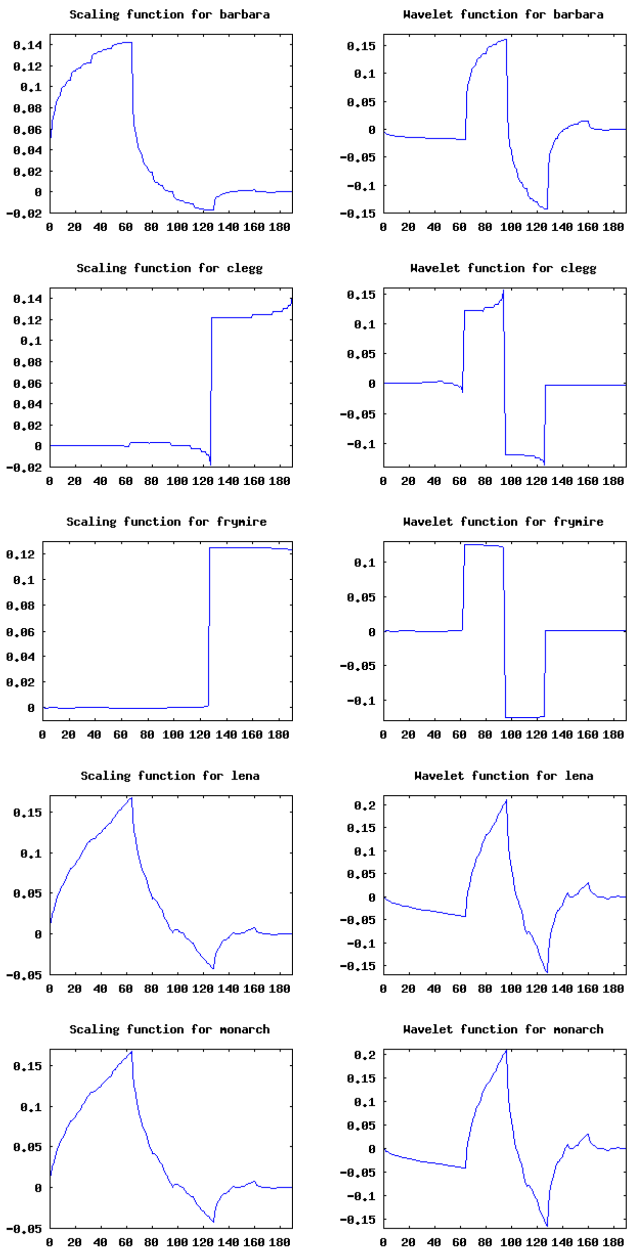


Fig. 3. Pairs of scaling (left) and wavelet (right) functions for the filters of Table 2. Each row: barbara, clegg, frymire, lena, and monarch.

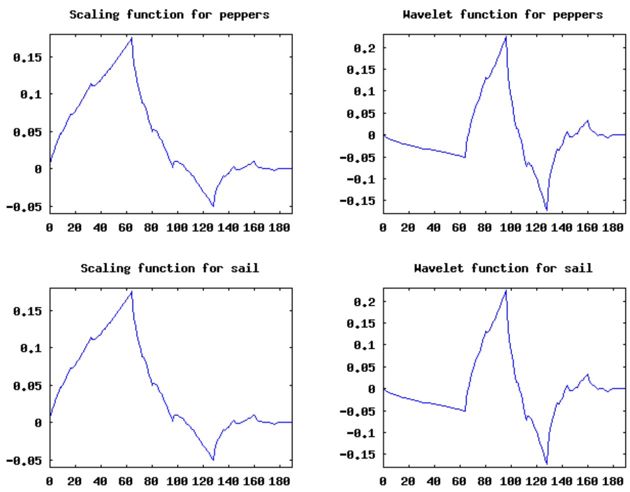


Fig. 4. Pairs of scaling (left) and wavelet (right) functions for the filters of Table 2. Each row: peppers, and sail.

the same image and different dimensions (128x128, 256x256 and 512x512 pixeles) and we note that the free parameter is very similar.

5 Conclusions and Future Work

We have shown experimentally that it is possible to get different filters for different signals adjusting the single free parameter to describe the four length h_k sequences that correspond to the QMF filters used to implement the discrete wavelet transform. These filters provide perfect reconstruction, and the experimental RMSE is zero in all cases. The function that measures the cumulative energy allows to guide the fitness of the individual of a genetic algorithm, hence the best of them increase the number of negligible coefficients and concentrate the energy, which is convenient for signal compression. However, the execution time suggests that this method, as is, cannot be used in real time applications. To compress a single image we requiere to calculate the equivalent to $2PG = 6560$ wavelet image transforms. We noticed that similar images give a similar value for the free parameter, so we consider that it can be used in applications of image classification. Future work will be focused on the reduction of the time execution as well as the required computational resources to implement it in hardware with limited resources. Also, we expect to design larger filters, to deal with wavelet packet based filters, to reduce the search time, and to work with automatic image classification.

References

1. Joseph, O., Chapa, M.: Algorithms for designing wavelets to match a specified signal. IEEE Transactions on signal processing 48(12), 3395–3406 (2000)
2. Golden, S.: Identifying Multiscale Statistical Models Using the Wavelet Transform, S. M. thesis. Dept. of Elect. Eng. and Comput. Sci., Mass. Inst. of Tech, Cambridge, MA (1988)
3. Herrera, O.: Lossless compression using metasymbols, Ph. D. thesis. Center for Computer Research, National Polytechnic Institute, Mexico (2005)
4. Daubechies, I.: Ten lectures on wavelets. SIAM, Philadelphia (1992)
5. Sheng, Y.: Wavelet transform in the transforms and applications handbook. In: Poularikas, H.D. (ed.) The Electrical Engineering Handbook Series, pp. 747–827. MCRC Press
6. Burrus, C., Gopinath, R., Guo, H.: Introduction to wavelets and wavelet transform, a primer. Prentice Hall, Upper Saddle River (1998)
7. Hehong, Z., Tewfik, A.: Parametrization of compactly supported orthonormal wavelets. IEEE Transactions on Signal Processing 41(3), 1428–1431 (1993)
8. Daubechies, I.: Orthonormal bases of compactly supported wavelets. Commun. on Pure and Applied Math. 41(1), 909–996 (1988)
9. Tewfik, A., Sinha, D., Jorgensen, P.: On the optimal choice of a wavelet for signal representation. IEEE Transactions on information theory 38(2), 747–765 (1992)
10. Walker, J.: A primer on wavelets and their scientific applications, 2nd edn. Chapman Hall/CRC, Taylor and Francis Group (2008)
11. Kuri, A.: A Comprehensive Approcah to Genetic Algorithms in Optimization and Learning. National Polytechnic Institute, Mexico (1999)
12. Mallat, S.G.: Multiresolution approximations and wavelet orthonormal bases of $\mathbb{L}^2\mathbb{R}$. Trans. Amer. Math. Soc. 315(1), 69–87 (1089)
13. Mallat, S.G.: A theory for multiresolution signal decomposition: The wavelet representation. IEEE Trans. on Patt. Anal. Mach. Intell. 11(7), 674–693 (1989)

Financial Time Series Prediction in Cooperating with Event Knowledge: A Fuzzy Approach

Do-Thanh Sang[1], Dong-Min Woo[1], Dong-Chul Park[1], and Thi Nguyen[2]

[1] Department of Electronics Engineering, Myongji University, Korea 449-728
[2] School of Geography and Environmental Science, Monash University, Australia
sang.dothanh@gmail.com

Abstract. A number of researchers have used historical numeric time series data to forecast financial markets, i.e. stock prices, and they achieved some results with reasonable accuracies. However, there are various non-numerical factors that influence prices such as company's performance, government involvement, trends of the market, changes in economic activity and so forth. We attempt to take such factors into account to our recent study. This paper surveys an application of a fuzzy inference system, namely Standard Additive Model, for predicting stock prices in cooperating with event-knowledge and several new training criteria. Experimental results show that the integrated model yields the outcomes which have error smaller than original model's one.

Keywords: Standard Additive Fuzzy System, financial time series prediction, event knowledge, fuzzy logic

1 Introduction

Stock market prediction has been a critical issue in the field of finance, mathematics and engineering due to its potential financial gain. Many have attempted to forecast the stock prices using various computational tools such as Support Vector Machine [1], Multiple Linear Regression [2], Hidden Markov Model [3], Neural Network (NN) and others [4]. NN is the most prominent technique over the last decade, however it still shows shortcomings. It is considered as a "black box" in which it is not handy to keep track adjustments of parameters occurring in the system. Instead, the inference fuzzy system is recommended [5]. A fuzzy system bases on fuzzy logic which has the advantage that the solution to the problem can be cast in terms that human operators can understand, so that their experience can be used in the design. On the other hand, fuzzy logic solutions are easy to verify and optimize.

There are many factors that can influence the share price. These factors can be derived from the news release about companies or superpower national economy. These incidents are called "events" [6]. The primary reason of incorporating event-knowledge (EK) in stock market prediction is based on an assumption that the future price of a stock partially depends on various political and international events as alongside the various economic indicators. Thus, many studies have used event information (qualitative factors) as well as quantitative data in predicting stock markets. One of the first studies using the EK was performed by Kohara et al. [7]. They incorporated EK extracted from the newspaper headlines and then made whether a particular event can positively influence the stock market tendencies or not. Hong and Han [8] introduced an automated system (KBNMiner) that acquires EK from the Internet for the prediction of interest rates. What is notable from their study is that the web mining technique they applied to predicting interest rates can also be used for stock price prediction. Nonetheless, extracting/collecting the historical events is not a trivial task. Yan et al. [9] showed an effective method to extract EK from the series data with an assumption that EK was reflected in that series.

Most applications of time series forecasting rely on the goodness-of-fit as their performance criterion. Regarding financial data, this criterion is not always appropriate. Yao and Tan [10] proposed a profit based adjusted weight factor for supervised training process. Instead of using the traditional least square error, they add a factor which contains the profit, direction, and time information to the error function. Their results show that this new approach does improve the predictability of neural network models, for the financial application domain. Another stopping criterion in training the system is the selective presentation learning (SEL) algorithm which was given by Kohara [11]. The training data is separated into large-change data and small-change data and large-change data is presented more often than small-change one.

This paper surveys the incorporation between Standard Additive Model (SAM) fuzzy system [12] incorporating with Genetic Algorithm (GA) using event-knowledge and new performance criteria to forecasting stock prices. The goal of this survey is to find a best model for financial time series prediction in terms of profit earning.

The paper is organized as follows. Two next sections describe briefly about the SAM fuzzy system and learning process. Section 4 presents the extracting EK from data. Section 5 explains new training criteria. Survey results are shown in section 6. Finally, our conclusion is summarized in the last section.

2 Standard Additive Model

A fuzzy inference system consists of m fuzzy rules (Fig. 1). Each fuzzy rule is a conditional IF-THEN proposition of the form R_j: IF x = A_j THEN y = B_j; $j = \overline{1,m}$; where $x \in R_n$, $y \in R_p$, are multi-dimensional vectors, A_j and B_j are fuzzy sets on the input space X and the output space Y respectively.

Because of SUM (summary) combination of fuzzy rules, this fuzzy system is named Standard Additive Model (SAM). In SAM, each input x fires j^{th} fuzzy rule and results in fuzzy set B_j' determined by the PRODUCT operation between membership degree of if-part a_j(x) in [0,1] and then-part fuzzy set $B_j : B'_j = a_j(x)B_j$. The system can use some factored form such as $a_j(x) = a^1_j(x_1)...a^n_j(x_n)$ or $a_j(x) = min(a^1_j(x_1), ..., a^n_j(x_n))$. Then-part fuzzy set $B_j \subset R_p$ has set function $b_j : R_p \to [0, 1]$ and volume V_j and centroid (center of gravity) c_j of fuzzy set.

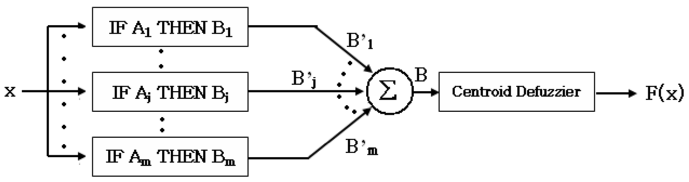


Fig. 1. SAM's parallel combination structure

The defuzzification method of SAM is CoG (Centroid of Gravity) of B:

$$F(x) = Centroid\left(\sum_{j=1}^{m} w_j a_j(x) B_j\right) = \frac{\sum_{j=1}^{m} w_j a_j(x) V_j c_j}{\sum_{j=1}^{m} w_j a_j(x) V_j} = \sum_{j=1}^{m} p_j(x) c_j \qquad (1)$$

where w_j is the corresponding weight of the j^{th} rule. The convex weight:

$$p_j\left(x\right) = \frac{w_j a_j\left(x\right) V_j}{\sum_{k=1}^{m} w_j a_k\left(x\right) V_k} \qquad (2)$$

The SAM fuzzy system can uniformly approximate continuous and bounded measurable functions on compact domains. If y = f(x) is not analytically known, we cannot write an equation in explicit form. However, we can use the relationship between the input space X and the output space Y which given by Y = F(x), a relationship that links subsets of the input space X to subsets of the output space Y by fuzzy terms $R_j = A_j \times B_j$. Fig. 2 illustrates the fuzzy rule patches in the input-output space and how these patches cover the graph of f(x).

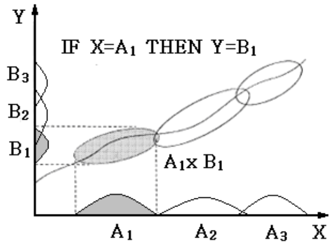


Fig. 2. Mapping of input space into output space in case of 2-D

2.1 Shape of Fuzzy Sets

The shape - membership function - of if-part fuzzy sets affects how well fuzzy systems approximate nonlinear continuous functions [13]. There is no function as best selection, it depends on each specific application. In trying to achieve accuracy and stability in application, all membership functions are examined in this research. Some membership functions (e.g. trapezoid, difference hyperbolic tangent, metrical difference logistic) often take much time in processing and therefore they are less effective.

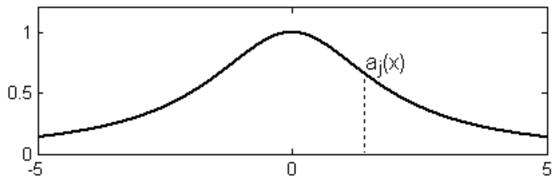


Fig. 3. *Cauchy* set function in 1-D case

Gaussian function gives richer fuzzy systems with simple learning laws that tune the bell-curve means and variances. But this popular choice comes with a special cost: it converts fuzzy systems to radial-basis-function neural networks or to other well-known systems that predate fuzzy systems. The *Cauchy* function is a bell curve with thicker tails than the Gaussian bell curve and with infinite variance and higher order moments. Thus the *Cauchy* set function is chosen for experiments in this research. The j^{th} *Cauchy* set function (Fig. 3) centered at m_j and width $d_j > 0$ is defined as

$$a_j(x) = \frac{1}{1 + \left(\frac{x - m_j}{d_j}\right)^2} \tag{3}$$

2.2 Constructing Fuzzy Rules

Based on the clustering results, the fuzzy rules system is constituted. The centers of fuzzy rules c_j are identified via centroid vectors. The width of j^{th} fuzzy set, for example, is set simply via its neighbors by the following formula

$$V_j = \frac{|c_j - c_c|}{r} \tag{4}$$

where c_j is the center of j^{th} fuzzy set, c_c is the center of the closest fuzzy set, and r is the overlap coefficient. Generally, a prediction problem is considered as one of non-linear function approximations wherein the function's approximated value will be used as the prediction. In order to improve approximating performance, SAM needs a strong learning process aiming at obtain a set of robust fuzzy rules. Through modifying volume and centroid of fuzzy rules, SAM relocates automatically fuzzy patches' position and size hereby the approximation can be expected more accurate. Regarding SAM, a learning method is evaluated whether well or not based on the way it ensures to maintain fuzzy patches at curve points in graphical presentation of the f(x) function.

3 Learning Process

The learning process of SAM (or fuzzy system, generally) usually encompasses two main stages those are structure (unsupervised) learning and parameter (supervised) learning as shown in Fig. 4. Also, optimal learning is employed to eliminate unnecessary fuzzy rules.

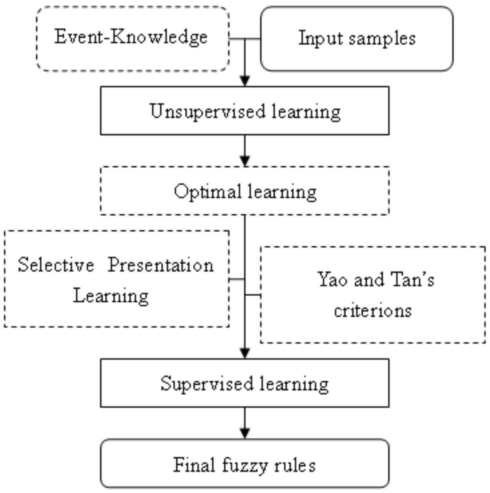


Fig. 4. Learning process of SAM in cooperating with EK and new training criteria

3.1 Structure Learning

Kohonen's Self Organizing Map (SOM) algorithm [14] has been applied in this research. SOM as an unsupervised learning algorithm is one of the most popular neural network algorithms and produces variation for improving its performance. The SOM can be summarized as

$$w_j[n+1] = w_j[n] + \alpha[n](x - w_j[n]) \tag{5}$$

where output neuron w_j is the winner neuron and neighbor neurons to the winner j at the iteration n. The learning coefficient $\alpha[n]$ at iteration n defines a decaying constant with an iteration such as

$$\alpha[n] = c(0)\left(1 - \frac{n}{N}\right) \tag{6}$$

with predetermined constant c(0) and total number of iterations N. The SOM holds the very advantageous preserving property that can capture the probability distribution density of the input data without help of external supervision.

After clustering data patterns, the next task is to build up fuzzy rules from their centroid vectors using fuzzy sets. Constructing fuzzy rules is stated in previous section.

3.2 Optimal Learning

Theoretically, regarding the fuzzy system in general or the SAM in particular, the more number of fuzzy rules the more accuracy in approximation process. Nevertheless, if a system has too many fuzzy rules, it would take a long time in learning process. An optimal system will only keep necessary fuzzy rules.
One of solutions for the above problem is using Genetic Algorithm (GA) [15]. The detailed GA for SAM fuzzy system is as follows:

– **Step 1:** Initialize ten chromosomes. Each chromosome is a chain of binary values describing status of corresponding rules in SAM. Value "*0*" means the rule is omitted while value "*1*" means the rule is selected. Every generation only uses ten chromosomes. One of individuals in the first generation contains all rules (all gene values of chromosome are equal to "*1*").
– **Step 2:** Create new chromosomes by crossover (probability 0.5) and mutation (probability 0.01).
– **Step 3:** Use roulette wheel with adaptive function to select ten best chromosomes which have the minimal Fit(.) value.

$$Fit(m) = ln\left(\bar{\sigma}_\varepsilon^2\right) + \frac{log_n(m)}{n} \tag{7}$$

where:

$$\bar{\sigma}_\varepsilon = \frac{1}{n}\sum_{j=1}^{n}(y_j - F(x_j))^2 \tag{8}$$

- m: number of used rules.
- n: number of training data samples.

– **Step 4:** If the stopping condition (i.e. expected error) is not satisfied, return step 2.
– **Step 5:** Choose the best one in ten chromosomes at the final population.

The found binary chain of the best chromosome will be used for eliminating unnecessary rules.

3.3 Parameter Learning

The gradient descent algorithm [16] has been deployed in this step. The aim of parameter adjustment learning phase is to minimize the square of error:

$$E(x) = \frac{1}{2}(f(x) - F(x))^2 \tag{9}$$

The learning rule applies for variable ξ in SAM has following form:

$$\xi(t+1) = \xi(t) - \mu_t\frac{\partial E}{\partial \xi} \tag{10}$$

where μ_t is the learning rate. The learning rule for each parameter is expanded in detail as follows.

$$\frac{\partial E}{\partial F} = -(f(x) - F(x)) = -\varepsilon(x) \tag{11}$$

$$c_j(t+1) = c_j(t) + \mu_t.\varepsilon(x).p_j(x) \tag{12}$$

$$V_j(t+1) = V_j(t) + \mu_t.\varepsilon(x).\left[c_j - F(x)\right].\frac{p_j(x)}{V_j} \tag{13}$$

$$w_j(t+1) = w_j(t) + \mu_t.\varepsilon(x).\left[c_j - F(x)\right].\frac{p_j(x)}{w_j} \tag{14}$$

Parameters of *Cauchy* membership function are tuned:

$$m_j(t+1) = m_j(t) + 2\mu_t\varepsilon(x)p_j(x)[c_j - F(x)]\frac{x - m_j}{d_j^2}a_j(x) \tag{15}$$

$$d_j(t+1) = d_j(t) + 2\mu_t\varepsilon(x)p_j(x)[c_j - F(x)]\frac{(x - m_j)^2}{d_j^3}a_j(x) \tag{16}$$

All initial configuration parameters are chosen based on training data set. The momentum technique is also integrated in the parameter tuning process. This helps the supervised learning to avoid local minimum cases and to reduce the learning time. The learning formula with momentum is as follows

$$\xi(t+1) = \xi(t) - \mu_t\frac{\partial E}{\partial \xi} + \gamma\Delta\xi(t) \tag{17}$$

where γ is the momentum coefficient.

4 Extracting Event-Knowledge

Knowledge of the events is divided into two kinds: negative-event-knowledge for events which tend to reduce stock prices and positive-event-knowledge for events which tend to raise them. Event-knowledge (EK) is non-deterministic knowledge showing only tendencies for stock price prediction. Actual stock price movement is influenced by a wide range of factors including numerical economic indicators and the EK. Therefore, the direction predicted using only EK does not always correspond to actual direction of next-day's stock price movement. In Kohara's experiments, the directions indicated by extracted EK corresponded to the actual next-day's direction approximately 60% of the time. The EK is stated abstractly by IF-THEN rules as follows:

Rule 1: IF (domestic politics = good) THEN (event = positive-event)
Rule 2: IF (domestic politics = bad) THEN (event = negative-event)
Rule 3: IF (business prospects = good) THEN (event = positive-event)
Rule 4: IF (business prospects = bad) THEN (event = negative -event)

A set of above rules could be expanded with more factors to constitute prior knowledge of system. The detailed extracting EK from time series algorithm includes following steps:
- **Step 1:** Make smooth series by single exponential technique [17].
- **Step 2:** Compute the difference values between smoothed and original values.

- **Step 3:** Normalize difference values into [0, 1].
- **Step 4:** Assign normalized value at (t+1) to EK value at (t).

Smoothing allows the model series-data to be represented as a really nonlinear function that is assumed to present the trend of time series in normal condition. The normalized difference value at (t + 1) shows the influence-degree of summarized events at (t) onto time series at (t + 1). The near 1 (one) value presents influence-degree of positive-EK whereas the value 0 (zero) represents the opposite case. An EK value of 0.5 presents the normal condition or the balance of influence degree between positive-EK and negative-EK. Application of EK into SAM fuzzy system is depicted in Fig 5.

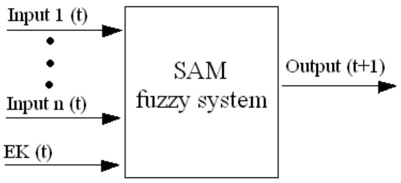


Fig. 5. EK(t) is fed to the system to predict value (t+1)

5 New Training Criteria

The traditional back-propagation training criterion is based on the Least Squares or Ordinary Least Squares (OLS) error function.

$$E_{OLS} = \frac{1}{2N} \sum_{p=1}^{N} (t_p - o_p)^2 \tag{18}$$

where N is number of data in training set. Just for convenience, we notate target value f(x) and output value F(x) in Equation 9 by t_p and o_p respectively from now on. New functions are defined in using concept of Selective Presentation Learning (SEL) and three criteria of Yao and Tan. They all are described in following subsections.

5.1 Selective Presentation Learning

In ordinary training, the number of presentations of all training data is usually independent of the size of the changes in the target time series. Learning process is usually stopped at the point of minimal mean squared error between the network's outputs and the actual outputs. The following two problems are considered.

Problem 1: Generally, the ability to predict large changes is more important than the ability to predict small changes. When all training data are presented equally as in conventional learning, we assume that the systems learn small changes as accurately as large changes and thus cannot reflect large changes more accurately.

Problem 2: The target function for forecasting problems is not always to minimize the average of prediction errors. Regarding financial market, maximizing profits through financial dealing by using predicted values is more significant than minimizing errors. Minimizing the average of prediction errors does not always correspond to maximizing profits.

Kohara [7] overcame these problems by separating the training data into large-change data and small-change data. Large-change data (small-change data) has next-day changes that are larger (smaller) than a preset value. Large-change data are presented to training process more often than small-change data. For instance, all training data are presented in the first learning cycle, only large-change data are presented in the second cycle, and so forth. The outline of the selective presentation learning algorithm is as follows.

- **Step 1:** Separate the training data into large-change and small-change sets.
- **Step 2:** Perform supervised learning with more presentations of large-change data than of small-change data.
- **Step 3:** Stop learning at the point satisfying a certain stopping criterion (e.g. stop at the point having the maximum profit).

5.2 Yao and Tan's Criteria

Yan and Tan added a factor which contains the profit, direction, and time information to the error function. The following research hypotheses were proposed:

Hypothesis 1: In addition to ordinary least squares error function, a factor representing the PROFIT could be added to the error function in order to improve the forecast ability of models. This model is called Directional Profit (DP) model. New profit adjust factor is a function of changes and direction.

$$f_{DP}(p) = F(|t_p - t_{p-1}|, sign(\Delta t_p, \Delta o_p)) \tag{19}$$

The Directional Profit adjustment factor is stated as

$$f_{DP}(p) = \begin{cases} a_1 \ \ if \ \Delta t_p * \Delta o_p > 0 \ and \ |\Delta t_p| \leq \sigma \\ a_2 \ \ if \ \Delta t_p * \Delta o_p > 0 \ and \ |\Delta t_p| > \sigma \\ a_3 \ \ if \ \Delta t_p * \Delta o_p < 0 \ and \ |\Delta t_p| \leq \sigma \\ a_4 \ \ if \ \Delta t_p * \Delta o_p < 0 \ and \ |\Delta t_p| > \sigma \end{cases} \tag{20}$$

where σ is a threshold of the changes of sample data; $a_1, ..., a_4$ are constant. The standard deviation of the training data set is used.

$$\sigma = \frac{1}{N} \sum_{p=1}^{N} (t_p - \mu) \tag{21}$$

where μ is the mean of the target series. The new error function will be

$$E_{DP} = \frac{1}{2N}\sum_{p=1}^{N} f_{DP}(p)(t_p - o_p)^2 \tag{22}$$

Hypothesis 2: A factor representing the TIME could be added to the error function in order to improve the forecast ability of models. This model is called Discounted Least Squares (DLS) model.

$$E_{DLS} = \frac{1}{2N}\sum_{p=1}^{N} w(p)(t_p - o_p)^2 \tag{23}$$

where $w(p)$ is the contribution of observation to the overall error

$$w(p) = \frac{1}{1 + e^{(a - \frac{2ap}{N})}} \tag{24}$$

Hypothesis 3: If PROFIT and TIME are useful factors of error function to improve the forecast ability, an even better result could be achieved by using the combination of both of them. This model is called Time dependent Directional Profit (TDP) model.

$$E_{TDP} = \frac{1}{2N}\sum_{p=1}^{N} f_{DP}(p) * w(p)(t_p - o_p)^2 \tag{25}$$

6 Experimental Results

We construct five models accompanied by event-knowledge (EK), namely EK Selective Presentation Learning (EK-SEL) model, EK Directional Profit (EK-DP) model, EK Discounted Least Squares (EK-DLS) model, EK Time dependent Directional Profit (EK-TDP) model and rough EK model which only incorporates EK with Ordinary Least Squares (OLS). They are benchmarked with traditional OLS model.

Five stock prices are tested in our study, they are iShares FTSE China Index (FCHI), Microsoft Corporation (MSFT), Dell Inc. (DELL), The Coca-Cola Company (KO) and Bank of America Corporation (BAC). All of them are downloaded from Yahoo Finance website http://finance.yahoo.com. The sampled period ranges up to 01 Jan 2010. The first day of each dataset is exhibited in Table 1. We choose 5% last data of series to be testing set and evaluate performance of each model in terms of Mean Absolute Error (MAE) on those.

The configuration of SAM fuzzy system in five datasets is identical. In detail, SAM uses 20 fuzzy clusters, learning rate is 0.0001, momentum coefficient is 0.9, number of generation in GA is 5 and training cycle is 4000 epochs. The dimension of an input pattern is 4, it means four historical values in time series

Table 1. Statistical results on five stock prices

	FCHI	MSFT	DELL	KO	BAC
Num. of data	252	401	505	756	1007
First day	02Jan 09	02Jun 08	02Jan 08	03Jan 07	03Jan 06
MAE of OLS	1.6319	0.4887	0.5639	0.4131	1.2227
MAE of EK	**0.8952**	0.3914	0.5566	0.3525	0.5214
MAE of EK-SEL	1.2025	**0.3249**	**0.3457**	**0.2209**	**0.4061**
MAE of EK-DP	1.3379	0.3901	0.4073	0.2357	0.431
MAE of EK-DLS	1.0944	0.3956	0.3654	0.3674	0.5537
MAE of EK-TDP	1.1722	0.3687	0.4606	0.2429	0.5669
Best model	EK	EK-SEL	EK-SEL	EK-SEL	EK-SEL

produce one predict value. This number is chosen by knowledge that stock price of current day is affected by last four days and not cause over-fitting while using large number. Note that with models which use EK, the dimension of an input pattern is also 4, but the last element is EK value.

From the statistical results in Table 1, it is easy to realize that MAEs of five new models are always smaller than MAEs in original model although the number of testing data is different. Model EK-SEL surpasses other models in four stock prices except FCHI. Criterion TDP was proved to achieve the performance better than DP and DLS [10], however, while used with EK, it does not show the excessive efficiency. The reason is stock prices in period end-half of 2009 were altered much more than usual and model SEL contains significant factors than TIME & PROFIT factors. Moreover, when we perform the supervised learning, EK-DLS and EK-TDP models cause unstable error function because of using Equation 24 with index factor p.

7 Conclusions

We have used event-knowledge and new training criteria to improve the ability to predict large changes by SAM fuzzy system. The results of several experiments on stock market prediction showed that model EK_SEL outperforms conventional approaches. We understand that success in several kind of stock prices does not necessarily mean the same for others and the financial market often comprises many complicated factors. Nevertheless, when data mining technique has been advanced rapidly recently, our approach could be seen as new research direction involves demands of economic forecasting.

References

1. Shuo, H., Chen, R.C.: Using SVM with Financial Statement Analysis for Prediction of Stocks. Communications of the IIMA 7(4), 63–72 (2007)
2. Ismail, Z., Yahya, A., Shabri, A.: Forecasting Gold Prices Using Multiple Linear Regression Method. American Journal of Applied Sciences 6(8), 1509–1514 (2009)

3. Hassan, M.R., Baikunth, N.: Stock Market Forecasting Using Hidden Markov Model: A New Approach. In: Proc. of ISDA, vol. 1, pp. 192–196 (2005)
4. Wei, S.: Theory Survey of Stock Yield Prediction Models. Journal of Economics and Finance 1(1), 175–182 (2009)
5. Ebrahim, A., Amir, A.: Stock Price Forecast by Using Neuro-Fuzzy Inference System. Journal of Business, Economics, Finance and Management Sciences 1, 216–219 (2009)
6. Ng, A., Fu, A.W.: Mining Frequent Episodes for Relating Financial Events and Stock Trends. In: Whang, K.-Y., Jeon, J., Shim, K., Srivastava, J. (eds.) PAKDD 2003. LNCS (LNAI), vol. 2637, pp. 27–39. Springer, Heidelberg (2003)
7. Kohara, K., Ishikawa, T., Fukuhara, Y., Nakamura, Y.: Stock Price Prediction Using Prior Knowledge and Neural Networks. Intelligent Systems in Accounting, Finance and Management 6(11), 11–22 (1997)
8. Hong, T., Han, I.: Integrated approach of cognitive maps and neural networks using qualitative information on the World Wide Web: KBN Miner. Expert Systems 21(5), 243–252 (2004)
9. Yan, X.B., et al.: Research on event prediction in time-series data. In: Proc. of Int. Conf. on Machine Learning and Cybernetics, vol. 5, pp. 2874–2878 (2004)
10. Yao, J.T., Tan, C.L.: A Study on Training Criteria for Financial Time Series Forecasting. In: Proc. of Int. Conf. on Neural Information Processing, pp. 772–777 (2001)
11. Kohara, K.: Neural Multivariate Prediction Using Event-Knowledge and Selective Presentation Learning. In: IEEE Int. Conf. on Neural Networks, vol. 1, pp. 359–364 (1995)
12. Kosko, B.: Fuzzy Engineering. Prentice Hall PTR, Englewood Cliffs (1996)
13. Nguyen, T.T., Peterson, J.: Enhancing the Performance of the Fuzzy System Approach to Prediction. In: Proc. of FSKD, vol. 1, pp. 259–265 (2008)
14. Kohonen, T.: Self-organizing maps. Springer, Heidelberg (2001)
15. Jong, R.K., Do, U.J.: Optimizing the fuzzy classification system through genetic algorithm. In: Proc. of ICCIT, vol. 2, pp. 903–908 (2008)
16. Robert, M.F.: The Steepest Descent Algorithm for Unconstrained Optimization and a Bisection Line-search Method. Massachusetts Institute of Technology (2004)
17. Cipra, T., McKenzie, J., McKenzie, E.: Exponential smoothing for irregular data. Journal of Applications of Mathematics 51(6), 597–604 (2006)

The Fuzzy Syllogistic System*

Bora İ. Kumova and Hüseyin Çakır

Department of Computer Engineering
İzmir Institute of Technology
35430 İzmir, Turkey
{borakumova,huseyincakir}@iyte.edu.tr

Abstract. A categorical syllogism is a rule of inference, consisting of two premisses and one conclusion. Every premiss and conclusion consists of dual relationships between the objects M, P, S. Logicians usually use only true syllogisms for deductive reasoning. After predicate logic had superseded syllogisms in the 19th century, interest on the syllogistic system vanished. We have analysed the syllogistic system, which consists of 256 syllogistic moods in total, algorithmically. We have discovered that the symmetric structure of syllogistic figure formation is inherited to the moods and their truth values, making the syllogistic system an inherently symmetric reasoning mechanism, consisting of 25 true, 100 unlikely, 6 uncertain, 100 likely and 25 false moods. In this contribution, we discuss the most significant statistical properties of the syllogistic system and define on top of that the fuzzy syllogistic system. The fuzzy syllogistic system allows for syllogistic approximate reasoning inductively learned M, P, S relationships.

Keywords: Syllogistic reasoning; fallacies; automated reasoning; approximate reasoning; human-machine interaction.

1 Introduction

Although syllogism were superseded by propositional logic [8] in the 19th century, they are still matter of research. For instance philosophical studies have confirmed that syllogistic reasoning does model human reasoning with quantified object relationships [2]. For instance in psychology, studies have compared five experimental studies that used the full set of 256 syllogisms [5], [12] about different subjects. Two settings about choosing from a list of possible conclusions for given two premisses [6], [7], two settings about specifying possible conclusions for given premisses [9], and one setting about decide whether a given argument was valid or not [10]. It has been found that the results of these experiments were very similar and that differences in design appear to have had little effect on how human evaluate syllogisms [5]. These empirically obtained truth values for the 256 moods are mostly close to their mathematical truth ratios that we calculate with our algorithmic approach [11].

* This research was partially funded by the grant project 2009-İYTE-BAP-11.

Although the truth values of all 256 moods have been analysed empirically, mostly only logically correct syllogisms were used for reasoning or modus ponens and modus tolens, which are generalisations of syllogisms [13]. Uncertain application environments, such as human-machine interaction, require adaptation capabilities and approximate reasoning [15] to be able to reason with various sorts of uncertainties. For instance, we know that human may reason purposefully fallacious, aiming at deception or trickery. Doing so, a speaker may intent to encourage a listener to agree or disagree with the speaker's opinions. For instance, an argument may appeal to patriotism, family or may exploit an intellectual weakness of the listener. We are motivated by the idea for constructing a fuzzy syllogistic system of possibilistic arguments for calculating the truth ratios of illogical arguments and approximately reason with them.

Firstly, the syllogistic system is discussed briefly, including its most significant statistical properties, followed by our main contribution, which is the fuzzy syllogistic system with its possible application for recognising fallacies and reasoning with them.

2 The Syllogistic System

A categorical syllogism can be defined as a logical argument that is composed of two logical propositions for deducing a logical conclusion, where the propositions and the conclusion each consist of a quantified relationship between two objects.

2.1 Syllogistic Propositions

A syllogistic proposition or synonymously categorical proposition specifies a quantified relationship between two objects. We shall denote such relationships with the operator ψ. Four different types are distinguished $\psi \in \{A, E, I, O\}$ (*Table 1*):

• A is universal affirmative:	All S are P
• E is universal negative:	All S are not P
• I is particular affirmative:	Some S are P
• O is particular negative:	Some S are not P

One can observe that the proposition I has three cases (a), (b), (c) and O has (a), (b), (c). The cases I (c) and O (c) are controversial in the literature. Some do not consider them as valid [3] and some do [14]. We have experimentally proven that including these cases, harmonically completes the symmetry of the statistical structures of the syllogistic system [11].

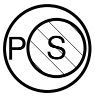
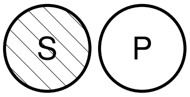
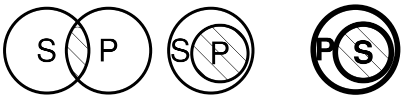
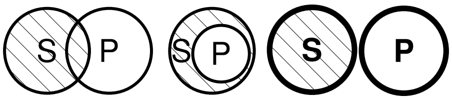
2.2 Syllogistic Figures

A syllogism consists of the three propositions major premise, minor premise and conclusion. The first proposition consist of a quantified relationship between the objects M and P, the second proposition of S and M, the conclusion of S and P (Table 2). Note the symmetrical combinations of the objects.

Since the proposition operator ψ may have 4 values, 64 syllogistic moods are possible for every figure and 256 moods for all 4 figures in total. For instance, AAA-1 constitutes the mood MAP, SAM - SAP in figure 1.

We shall denote a propositional statement with Φ_1, in order to distinguish between possibly equal propositional operators of the three statements of a particular mood, where i∈{1, 2, 3}.

Table 1. Syllogistic Propositions Consist of Quantified Object Relationships

Operator ψ	**Proposition** $\Phi(\psi)$	**Set-Theoretic Representation of Logical Cases***
A	All S are P	 $\lvert\Phi(A)\rvert=2$
E	All S are not P	 $\lvert\Phi(E)\rvert=2$
I	Some S are P	 (a)=3 (b)=2 **(c)**=2 (a)+(b)+(c)= $\lvert\Phi(I)\rvert=7$
O	Some S are not P	 (a)=3 (b)=2 **(c)**=2 (a)+(b)+(c)= $\lvert\Phi(O)\rvert=7$

* Number of sub-sets of a case (a), (b), (c) and total number of sub-sets of a proposition $\lvert\Phi(\psi)\rvert$.

2.3 Statistics About the Syllogistic System

The algorithm that we have introduced earlier [11] enables revealing various interesting statistics about the structural properties of the syllogistic system [4]. The most significant once are as follows.

First we calculate the truth values for every mood in form of a truth ration between its true and false cases, so that the truth ratio becomes a real number, normalised within [0.0, 1.0]. Thereafter we sort all moods in ascending order of their truth ratio (*Fig 1*). Note the symmetric distribution of the moods according their truth values. 25 moods have a ratio of 0 (false) and 25 have ratio 1 (true), where each is 25/256 = % 10.24 of all moods. 100 moods have a ratio between 0 and 0.5 and 100 have between 05 and 1, where each is 100/256 = % 0.390625. 6 moods have a ratio of exactly 0.5, which is % 0.0234375 of all moods.

For any three set, like M, P, S, in total 41 distinct intersections can be drawn. The 256 moods have in total 2624 truth cases, which map those 41 intersections multiple times. These mapping structures are also inherently symmetric. A complete discussion of all statistical details is presented in [4].

Table 2. Syllogistic Figures

Figure Name	I	II	III	IV
Major Premise	MψP	PψM	MψP	PψM
Minor Premise	SψM	SψM	MψS	MψS
Conclusion	SψP	SψP	SψP	SψP

3 Fuzzy Syllogistic Reasoning

Based on the symmetrical properties of the syllogistic system, we now define the fuzzy syllogistic system and a sample application for recognising fallacies.

3.1 Fuzzy Syllogistic System

From the structural properties of the syllogistic system [4], we elaborate now a fuzzified syllogistic system.

One can see (*Fig 1*) that every syllogistic case is now associated with one truth ration. We utilise the symmetric distribution of the truth ratios, for defining the membership function FuzzySyllogisticMood(x) = {CertainlyNot; Unlikely; Uncertain; Likely; Certainly} with a possibility distribution that is similarly symmetric (*Fig 1*). the possibility distribution of FuzzySyllogisticMood that was presented earlier, has been adapted to the values of the moods, such that moods with equal values have now equal linguistic values. The linguistic variable was adopted from a meta membership function for a possibilistic distribution of the concept likelihood [16]. The complete list with the names of all 256 moods is appended (*Table A1*).

As we have mentioned earlier, the algorithmically calculated truth ratios of the 256 moods (*Fig 1*) mostly comply with those empirically obtained truth ratios in psychological studies [5]. Hence the suggested possibilistic interpretation should reflect an approximately correct model of the syllogistic system.

3.2 Fuzzy Syllogistic Reasoning

Our objective is to design a new model for automated reasoning, which uses the fuzzy syllogistic system as reasoning mechanisms. For this purpose, we specify following methodology:

- Inductively accumulate sample instances of relationships between the objects M, P, S and classify them into the above mentioned 41 distinct sub-sets.
- Calculate the truth values of the 256 moods for these M, P, S relationships.
- Based on the cardinalities of the 41 sub-sets, calculate possible fallacies.
- Fuzzy syllogistic reason with the mood that has the highest truth value.

Fallacies may be identified manually, by a human, who is deciding on the proper semantics of the M, P, S relationships. However, in this methodology, we identify fallacies fully automated, based on the cardinalities of the sample 41 sub-sets.

3.3 Fallacies in Categorical Syllogisms

In logic, a fallacy is a misconception resulting from incorrect reasoning in argumentation. 7 fallacies are known in the literature for categorical syllogisms:

- Equivocation fallacy or fallacy of necessity: Unwarranted necessity is placed in the conclusion, by ignoring other possible solutions.
- Fallacy of undistributed middle: Middle term not distributed in at least one premiss.
- Illicit major/minor: Major/minor term undistributed in major/minor premiss, respectively, but distributed in the conclusion.
- Fallacy of exclusive premisses: Both premisses negative.
- Affirmative conclusion from negative premiss: Positive conclusion, but at least one negative premiss.
- E xistential fallacy: Both premisses universal, but particular conclusion.

These fallacies comply exactly with the 7 rules for eliminating invalid moods, which were discovered already by Aristotle [1].

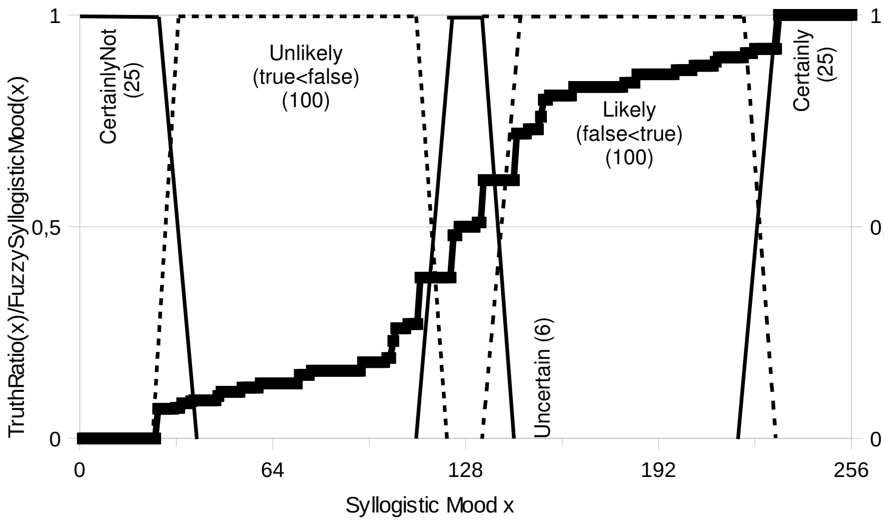


Fig. 1. 256 syllogistic moods sorted in ascending order of their TruthRatio(x) true/false, if number of truth cases of a mood is true<false and 1-false/true ratio, if false<true. Definition of the possibility distribution FuzzySyllogisticMood(x) with the linguistic variables CertainlyNot, Unlikely, Uncertain, Likely, Certainly and their cardinalities 25, 100, 6, 100, 25, respectively

3.4 Recognising Fallacies: Procedure

Our objective is to use the whole set of 256 syllogistic moods as one system of possibilistic argument for recognising fallacies and reasoning with them. For that purpose, we specify the following steps:

1. Calculate all truth cases and the truth ratio of a given mood.
2. Try to recognise fallacies with following rules, for identifying
   (a) possible false instances: reduction of A to I
   (b) possible true instances: reduction of E to O
   (c) further possible true instances: generalisation of I to A
   (d) further possible false instances: generalisation of O to E
   (e) complementing false instances: complementation of I to O
   (f) complementing true instances: complementation of O to I
3. Try to map the initial mood x to any mood y with a truth ratio closer to 1: TruthRatio(x) < TruthRatio(y)
4. Approximately reason with the truth ratios.

Rules (a)-(f) are generalisations of the above discussed reduction and conversion techniques.

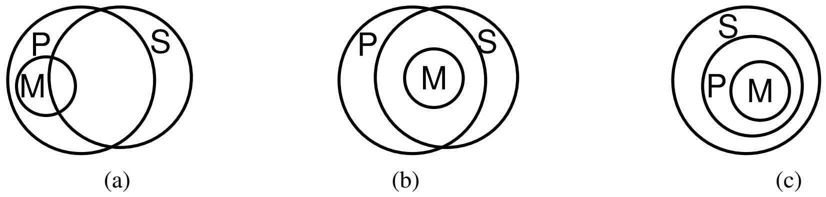


Fig. 2. False syllogistic cases of the mood AIA-1

3.5 Recognising Fallacies: Sample Application

We will now discuss these steps experimentally on the following example (*Fig 4*).

Firstly, we calculate the 3 true (*Fig 3*) and 3 false (*Fig 2*) cases of mood AIA-1 and its truth ratio of 0.5.

Secondly, we identify following fallacies:

- rule (a): Not all stories in The Child's Magic Horn (TCMH) are sad $\neg\Phi_1$(A). The truth is that only some stories in TCMH are sad Φ_1 (I).
- rule (a): Not all stories I cry at are stories in TCMH, because I will possibly cry at some other stories as well $\neg\Phi_3$(A). The truth is that only some of all the stories I cry at are stories in TCMH Φ_3(I).

Based on the identified fallacies and reductions Φ_1(A) to Φ_1(I) and Φ_3(A) to Φ_3(I), we can easily calculate the mood III-1 to be "more true" for the given sample propositions. In dead, mood III-1 has with 4 false/19 true cases 1-0.21=0.79, a better truth ratio.

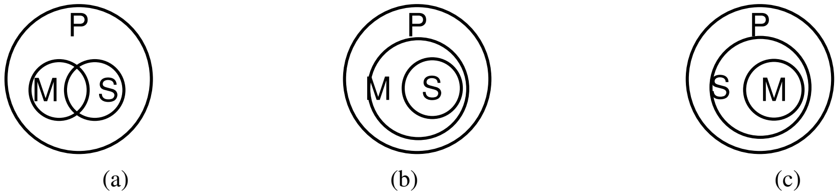


Fig. 3. True syllogistic cases of the mood AIA-1

P:	Sad	A: all M are P
M:	Stories in The Child's Magic Horn	I: some S are M
S:	Tales I cry at	
		A: all S are P

Φ_1(A):	All "Stories in The Child's Magic Horn" are "Sad"
Φ_2(I):	Some "Tales I cry at" are "Stories in The Child's Magic Horn"
Φ_3(A):	All "Tales I cry at" are "Sad"

Fig. 4. Sample syllogistic inference with the mood AIA of the syllogistic figure 1 (AIA-1)

Using mood III-1, we can now try to recognise further fallacies, by applying all combinations for complementing true or false possibilities, which yields the moods:

- OII-1: 1-7/17 = 0.59
- OOI-1: 1-7/17 = 0.59
- IIO-1: 1-6/17 = 0.65
- IOO-1: 1-5/17 = 0.71
- IOI-1: 1-5/17 = 0.71
- III-1: 1-4/19 = 0.79
- OOO-1: 1-3/21 = 0.86
- OIO-1: 1-3/21 = 0.86

Thirdly, these are 8 further candidates for replacing the initial mood AIA-1: 3/3 = 0.5. We may now chose OOO-1 or OIO-1, since both have equal truth ratio of 0.86.

In the last step, we may use the truth rations of the moods for fuzzy syllogistic reasoning as a model for approximate reasoning with quantified propositions.

3.6 Discussion

In the initial example (*Fig 4*), one can suspect possible fallacies in the positive generalisations Φ_1(A) and Φ_3(A), by intuitively assuming possible false instances in them.

The moods OOO-1 and OIO-1 have higher truth ratios than the initial mood $0.5 < 0.86$. Consider now the case OII-1 = 0.86

- Φ_1(O): Some "Stories in TCMH" are not "Sad"
- Φ_2(O): Some "Tales I cry at" are not "Stories in TCMH"
- Φ_3(O): Some "Tales I cry at" are not "Sad"

and OOI-1 = 0.86

- Φ_1(O): Some "Stories in TCMH" are not "Sad"
- Φ_2(I): Some "Tales I cry at" are "Stories in TCMH"
- Φ_3(O): Some "Tales I cry at" are not "Sad"

Although humans usually get confused from multiple existentially quantified propositions, we mostly assume intuitively that they are usually correct cases for reasoning, ie that they should have truth ratio close to 1.0. Nevertheless, these moods are mathematically not fully correct, as their truth ratios are considerably below 1.0. Now consider the case OOII-1 = 0.59

- Φ_1(O): Some "Stories in TCMH" are not "Sad"
- Φ_2(I): Some "Tales I cry at" are "Stories in TCMH"
- Φ_3(I): Some "Tales I cry at" are "Sad"

Usually, anyone will assume that this mood, like the both previous moods, is a correct case for reasoning. Although, their truth ratios differ with 0.86 – 0.59 = 0.27 considerably within the value range [0.0, 1.0]. This experimentally proves, what was known since medieval time, that humans tend to assume that reasoning with existential quantifiers are mostly confusion, but possibly correct. Possibly, because humans fail to combine multiple such fuzzy propositions logically correct. We can explain this phenomenon with the possible sub-sets of the propositions $|\Phi(A)| = 2$, $|\Phi(E)| = 2$, $|\Phi(I)| = 7$ and $|\Phi(O)| = 7$ (*Table 1*). Any figure including solely A or E propositions will have 6 sub-sets in total. Any figure including solely I or O propositions will have 21 sub-sets in total. Deciding about the correctness of a particular example requires approving or disapproving the truth of every single sub-set. Thus, propositions that consist of multiple existential quantifications are "too fuzzy" for humans to be decided logically correctly. However, as soon as at least several true sub-sets exist, humans tend to assume that the whole syllogism should be correct.

Finally, consider the mood AII-1 = 1.0

- Φ_1(A): All "Stories in TCMH" are "Sad"
- Φ_2(I): Some "Tales I cry at" are "Stories in TCMH"
- Φ_3(I): Some "Tales I cry at" are "Sad"

Assuming that Φ_1(A) is really true, ie M is a real sub-set of P, then we get a tautology. However, if we always strictly apply the above rules for recognising fallacies, then we should be able to identify almost always possible true or false instances within a given proposition. Hence, tautologies should rather be rare cases in real life.

4 Conclusion

Our algorithmic approach for calculating the truth ratios of syllogisms has enabled us to reveal all structural properties of the complete syllogistic system. On top of the syllogistic system we have proposed a fuzzy syllogistic system that consists of possibilistic arguments, which we have used in a sample application for recognising fallacies and fuzzy syllogistic reasoning with them.

We believe that this approach may prove a practical approach for reasoning with inductively learned knowledge, where P, M, S object relationships can be learned inductively and the "most true" mood can be calculated automatically for those relationships. That shall be our future work, alon with examples including recognising intentional or unintentional fallacies, with the objective to facilitate automated human-machine interaction.

References

[1] Aristotle.: The Works of Aristotle, vol. 1. Oxford University Press, Oxford (1937)
[2] Geurts, B.: Reasoning with quantifiers; Department of Philosophy; University of Nijmegen (2002)
[3] Brennan, J.G.: A Handbook of Logic. Brennan Press (2007)
[4] Çakır, H., Kumova, B.İ.: Structural Analysis of the Syllogistic System. In: International Conference on Fuzzy Computation (ICF 2010), Valencia/Spain (2010)
[5] Chater, N., Oaksford, M.: The probability heuristics model of syllogistic reasoning. Cognitive Psychology 38, 191–258 (1999)
[6] Dickstein, L.S.: The effect of figure on syllogistic reasoning. Memory and Cognition 6, 76–83 (1978)
[7] Dickstein, L.S.: Conversion and possibility in syllogistic reasoning. Bulletin of the Psychonomic Society 18, 229–232 (1981)
[8] Frege, L.G.F.: Begriffsschrift, eine der Arithmetischen Nachgebildete Formalsprache des Reinen Denkens. Verlag von Louis Nebert (1879)
[9] Johnson-Laird, P.N., Steedman, M.: "The psychology of syllogisms. Cognitive Psychology 10, 64–99 (1978)
[10] Johnson-Laird, P.N., Bara, B.G.: Syllogistic inference. Cognition 16, 1–61 (1984)
[11] Kumova, B.İ., Çakır, H.: Algorithmic Decision of Syllogisms. In: García-Pedrajas, N., Herrera, F., Fyfe, C., Benítez, J.M., Ali, M. (eds.) IEA/AIE 2010. LNCS, vol. 6097, pp. 28–38. Springer, Heidelberg (2010)
[12] Oaksford, M., Chater, N.: The probabilistic approach to human reasoning. Trends in Cognitive Sciences 5, 349–357 (2001)
[13] Russell, S., Norvig, P.: Artificial Intelligence - A Modern Approach. Prentice-Hall, Englewood Cliffs (2009)
[14] Wille, R.: Contextual Logic and Aristotle's Syllogistic. Springer, Heidelberg (2005)
[15] Zadeh, L.A.: Fuzzy Logic and Approximate Reasoning. Syntheses 30, 407–428 (1975)
[16] Zadeh, L.A., Bellman, R.E.: Local and fuzzy logics. In: Dunn, J.M., Epstein, G. (eds.) Modern Uses of Multiple-Valued Logic, Reidel, Dordrecht (1977)

Appendix A: Truth Degree of Syllogistic Moods

The table (*Table A1*) shows the x = [1, 256] moods in 5 categories with TruthRatio(x) normalised in [0.0, 1.0]. False and true moods are sorted according their number of false and true cases, respectively. Unlikely and Likely moods are sorted in ascending order of their truth ratio. The table also shows the possibility distribution of the membership function FuzzySyllogisticMood(x) $\in$ {CertainlyNot, Unlikely, Uncertain, Likely, Certainly}.

Table A1. Possibility Distribution FuzzySyllogisticMood(x) over the Syllogistic Moods in Increasing Order of Truth Ratio of the Mood x

Linguistic Value	**Sum**	**Mood x**
CertainlyNot; false; ratio=0	25	EIA-1, EIA-2, EIA-3, EIA-4, AIE-1, AIE-3, IAE-3, OAA-3, IAE-4, AOA-2, AAE-3, EAA-3, EAA-4, AAE-1, AAO-1, EAA-1, EAI-1, AEA-2, AEI-2, EAA-2, EAI-2, AAA-4, AAE-4, AEA-4, AEI-4
Unlikely; rather false; 0<ratio<0.5	100	EIE-1, IEE-1, EIE-2, IEE-2, EIE-3, IEE-3, EIE-4, IEE-4, AOE-2, OAA-2, OAE-2, AOA-1, IAA-1, OAE-1, OEE-1, IAA-2, EOE-3, OEE-3, AOE-4, EOE-4, OOE-3, AEA-1, AEE-1, AAA-3, AEA-3, AEE-3, EAE-3, EAE-4, EOE-1, EOE-2, OEA-2, OEE-2, OEA-4, OEE-4, OIE-1, OOE-1, OOA-4, OOE-4, IOA-3, IOE-3, OIE-3, IOA-4, IOE-4, IEA-1, IEA-2, IEA-3, IEA-4, IIA-1, IIA-2, IIA-3, IIA-4, IAE-1, OAA-1, OEA-1, AIE-2, IAE-2, OEA-3, AIE-4, AAA-2, AAE-2, EAA-1, EEE-1, EEA-2, EEE-2, EEA -3, EEE-3, EEA-4, EEE-4, IOA-1, IOE-1, IOA-2, IOE-2, OIA-2, OIE-2, OIA-4, OIE-4, OOA-2, OOE-2, OOA-3, IIE-1, IIE-2, IIE-3, IIE-4, AOE-3, IAA-3, OAE-3, IAA-4, OOA-1, OIA-1, OIA-3, AOE-1, AIA-2, EOA-3, AIA-4, AOA-4, EOA-4, OAA-4, OAE-4, EOA-1, EOA-2
Uncertain; undecided; ratio=0.5	6	AIA-1, AIO-1, AIA-3, AIO-3, AOA-3, AOO-3
Likely; rather true; 0.5<ratio<1.0	100	EOO-1, EOO-2, OIO-1, OOO-1, OIO-3, AIO-2, EOO-3, AIO-4, AOI-1, AOO-4, EOO-4, OAI-4, OAO-4, IAO-3, IAO-4, OAI-3, AOI-3, III-1, III-2, III-3, III-4, OOO-3, OOI-2, OOO-2, IOI-1, IOO-1, OII-2, OIO-2, IOI-2, IOO-2, OII-4, OIO-4, IAI-1, OAO-1, OEO-1, AII-2, OEO-3, IAI-2, AII-4, AAI-2, AAO-2, EEI-2, EEO-2, EEI-3, EEO-3, EEI-4, EEO-4, EEI-1, EEO-1, IIO-1, IIO-2, IIO-3, IIO-4, IEO-1, IEO-2, IEO-3, IEO-4, OII-1, OOI-1, IOI-3, IOO-3, OII-3, IOI-4, IOO-4, OOI-4, OOO-4, EOI-1, EOI-2, OEI-4, OEI-2, OEO-2, OEO-4, AEI-1, AEO-1, AAO-3, AEI-3, AEO-3, EAI-3, EAI-4, OOI-3, AOO-1, IAO-1, OAI-1, OEI-1, IAO-2, EOI-3, OEI-3, AOI-4, EOI-4, AOI-2, OAI-2, OAO-2, IEI-1, EII-1, EII-2, IEI-2, EII-3, IEI-3, EII-4, IEI-4
Certainly; true; ratio=1.0	25	EIO-1, EIO-2, EIO-3, EIO-4, AII-1, AII-3, IAI-3, OAO-3, IAI-4, AOO-2, AAI-3, EAO-3, EAO-4, AAA-1, AAI-1, EAE-1, EAO-1, AEE-2, AEO-2, EAE-2, EAO-2, AAI-4, AAO-4, AEE-4, AEO-4

Big Five Patterns for Software Engineering Roles Using an ANFIS Learning Approach with RAMSET

Luis G. Martínez, Antonio Rodríguez-Díaz, Guillermo Licea, and Juan R. Castro

Universidad Autónoma de Baja California
Calzada Tecnológico 14418, Tijuana, México 22300
{luisgmo,ardiaz,glicea,jrcastro}@uabc.edu.mx

Abstract. This paper proposes an ANFIS (Adaptive Network Based Fuzzy Inference System) Learning Approach where we have found patterns of personality types using Big Five Personality Tests for Software Engineering Roles in Software Development Project Teams as part of RAMSET (Role Assignment Methodology for Software Engineering Teams) methodology. An ANFIS model is applied to a set of role traits resulting from Big Five personality tests in our case studies obtaining a Takagi-Sugeno-Kang (TSK) Fuzzy Inference System (FIS) type model with rules that helps us recommend best suited roles for performing in software engineering teams.

Keywords: Fuzzy Logic, Uncertainty, Software Engineering, Psychometrics.

1 Introduction

Nowadays more and more mind-mapping and decision making programs find their way to our life. Use of effective decision making software for everyday planning and task management is on the rise in modern organizations. Web pages and Decision Groups commonly offer services for decision making and personnel selection with different methodologies applying psychometrics.

Diverse personality tests like Jung, Myers-Briggs, Keirsey, Big Five, among other projective tests, can be used to know the sociopsychological characteristics and personality of individuals besides abilities for job placement and hiring, especially in assigning individuals to form a working team [1][2][3].

Effective use of psychometric instruments can add value to an organization. When used in selection and structured interview process, they enable companies to select more accurately those people who will perform best in a role.

This paper proposes an ANFIS (Adaptive Network Based Fuzzy Inference System) Learning Approach to find personality type patterns of software engineering roles using Big Five Personality Tests while implementing RAMSET (Role Assignment Methodology for Software Engineering Teams), a personality based methodology used in software project development case studies. Personality tests are based on interpretation; therefore to tackle uncertainty a Takagi-Sugeno-Kang (TSK) Fuzzy Inference System (FIS) type model with rules will help us recommend best suited roles for performing in software engineering teams.

The rest of the paper is organized as follows: section 2 is a brief background of personality and personnel selection relationships. Section 3 defines RAMSET's steps and describes big five factors in general. Section 4 displays results of the big five personality test, the big five patterns and implementation of ANFIS model for RAMSET, concluding in section 5 with observations obtained from experience.

2 Background

Human Psychology studies are old, before the 80's personality aspects were considered of low value in personnel selection. Personnel selection methodologically chooses individuals for the right job. The first tests [4] used for a programmer's performance prediction and personnel selection in software development projects where PAT test (Programmer Aptitude Test), WPT test (Wonderlic Personnel Test) and PMA test (Primary Mental Abilities). In personality analysis, different and popular tests exist, like Jung's, Keirsey and Myers-Briggs Type Indicator (MBTI) tests. Rutherford [5] manages Personality Inventories in his Software Engineering classes to integrate heterogeneous teams, obtaining good results in teams' performance and individuals' growth. Karn and Cowling [6] also implemented personality tests in Sheffield University England, documenting personality effects in Software Engineering Teams performance, mentioning that is a vast area and many subjects to consider relating software engineer's personality and their teams.

Feldt et al. [7] corroborate that empirical studies should collect psychometrics focusing in correlating personality ant attitudes to software engineering processes and tools. Personality type was measured with MBTI (Myers-Briggs Type Indicator) using Keirsey Temperament Sorter by Gorla and Lam [8], their survey study shows a relationship between Software Engineering Roles and MBTI dimensions, primarily analyzing team leaders, analysts and programmers. These experiences resemble the findings by Capretz [9] in surveying people in the US working in software engineering.

Shen [10] and colleagues have applied Myers-Briggs Type Indicator (MBTI) and Kiersey Temperament Sorter to form engineering design teams based on Wilde's [11] own method of selection with this tests. They recommend a Sensing-Intuitive (SN) type as the most important preferred type linked to creativity for selection of engineers in a leader role for design process. Also Sodiya et al. [12] from Nigeria have developed a tool integrating personality traits proposing a Cogno-Personality Assessment Model for Software Engineering (CPAMSE) relating the Big Five Traits with roles and activities of engineers in software engineering.

In contemporary psychology, the "Big Five" factors of personality are five broad domains or dimensions of personality which have been scientifically discovered to define human personality. The initial model was reported by Tupes and Cristal [13], Digman [14] advanced his model and Goldberg [15] extended it to the highest level of organization, and it is known as the "Five Factor Model" or FFM [16], is a purely descriptive model of personality. The big five personality tests are used to screen candidates for team building, selection, job analysis, training programs, coaching, counseling and leadership development.

Personnel selection and assessment applies the measurement of individual differences to the hiring of people into jobs where they are likely to succeed. Industrial and organizational psychologists who practice in this area use information about the job and the candidates to help a company determine which candidate is most qualified for the job. Job analysis is the process of determining knowledge, skills, abilities and personal characteristics required for a given job. Based on results of job analysis, industrial and organizational psychologists choose selection methods which are most likely to be correlated with performance for the specific job.

Wei-Shen and Chung-Chian [17] proposed a realistic personnel selection tool based on fuzzy data mining method feasible to assist businesses in finding eligible applicants through reliable information effectively and efficiently. Therefore agreeing that prediction of future organizational behavior of employees should be the focal point in the processes of personnel selection, finding the relationship between the attributes applicants owned and organizational behaviors. Their tool can assist business manager to find eligible talent more efficiently. Their method is based on three types of behavior for a well-functioning organization: people must be induced to enter and remain with the organization; they must reliably carry out specific role or job requirements; and there also needs to be innovative and spontaneous activity that goes beyond role prescription [18].

In Software Engineering Development Teams each member can take a different role, for our case studies we adopted those defined by Tomayko [19]: architect, responsible for project creation, coordination and supervision; analyst, responsible for finding and following up on resources, requirement analysis and specifications; developer-programmer, responsible for implementation and code design; tester, responsible for tests and evaluation of the system; document specialist, responsible for compiling every document for evidence and defining documentation standards; and image and presentation role as a representative in charge of selling and promoting the product.

As we can see Software Engineers have their typical fields of expertise like coordinating, designing, programming, testing, evaluating, RAMSET methodology focus' is to determine the best suited role for team performance relating personality test results with software engineering roles. We can choose from a set of different Personality Tests all revolving on the dimension types or personality traits that display the behavior of the individual member of the team. Combining these tests we can acquire the most valuable information for decision making in assignment of roles.

This paper specifically analyzes results of internet's free Big Five Test applied in our different case studies with RAMSET methodology.

3 Methodology

At the University of Baja California, Tijuana Mexico teaching of Software Engineering in our Computer Engineering Program is being conducted with development of real software projects applying RAMSET: a Role Assignment Methodology for Software Engineering Teams based on personality, what is unique about our methodology is a combination of Sociometric techniques, Psychometrics and Role Theory in Software Engineering Development Projects, this methodology consists of the next steps:

a) Survey for abilities and skills.
b) Implementation of Personality Tests.
c) Execute Personal Interviews.

d) Implementation of the Sociometric Technique.
e) Assignment of Team Roles.
f) Follow up of Team Role fulfillment.

RAMSET methodology begins with a student's survey enumerating related courses of software engineering he has taken, to know which programming languages and data base managers he is expert in. The next step is a series of personality tests; they could be Jung, MBTI, Big Five, Keirsey or similar tests.

Subsequently we make an informal interview to know different aspects of his personality, what he likes to do, how he perceives himself at the end of his career studies, how he develops in the real world individually and with others, how he would like to participate in a team. After that a sociogram technique is applied to identify affinity for integration of teams. Based on test results and interview information a team role is recommended to the instructor so individual members of each team develop a specific team role with all its functions.

Big Five personality tests claim to measure your intensities in relation to the "Big Five" factors. The structure of the tests requires selecting options from multiple choice questionnaires. These big five personality tests equate your personality to your collective degrees of behavior in five factors.

The Big Five factors are Openness, Conscientiousness, Extroversion, Agreeableness, and Neuroticism (OCEAN, or CANOE if rearranged). The Neuroticism factor is sometimes referred to as Emotional Stability. And Openness factor sometimes is referred as Intellect.

Openness (O) is a disposition to be imaginative, inventive, curious, unconventional and autonomous, has an appreciation for art, emotion, adventure, unusual ideas, curiosity and variety of experience.

Conscientiousness (C) comprises of two related facets achievement and dependability, has a tendency to show self-discipline, be efficient, organized, act dutifully and aim for achievement, plans rather than behave spontaneously.

Extroversion (E) represents tendency to be sociable, outgoing and assertive, experiences positive affect such as energy, passion and excitement.

Agreeableness (A) is a tendency to be trusting, friendly, compassionate, cooperative, compliant, caring and gentle.

Neuroticism (N) represents tendency to exhibit poor emotional adjustment and experience negative or unpleasant emotions easily, such as anxiety, insecurity, depression and hostility.

Because of uncertainty of personality traits a fuzzy based approach is considered to provide an integrated quantity measure for abilities of software development personnel which incorporates all aspects of personality traits involved for role assignment.

Fuzzy based approaches have been considered like Lather's [20] fuzzy model to evaluate suitability of Software Developers, also Ghasem-Aghaee and Oren's [21][22] use of fuzzy logic to represent personality for human behavior simulation. Consequently encouraging engineering educators to make greater use of type theory when selecting and forming engineering design teams and delegating team roles, in benefit of achieving productivity and efficiency in team performance.

The neuro-adaptive learning method works similarly to that of neural networks. This method has been applied in pattern recognition in areas like 3D object recognition[23], fingerprint matching[24] and human facial expression recognition[25].

Neuro-adaptive learning techniques provide a method for the fuzzy modeling procedure to “learn” information about a data set. Fuzzy Logic Toolbox software computes the membership function parameters that best allow the associated fuzzy inference system to track the given input/output data. The Fuzzy Logic Toolbox function that accomplishes this membership function parameter adjustment is called ANFIS.

The acronym ANFIS derives its name from Adaptive Neuro-Fuzzy Inference System as defined by Jyh-Shing Roger Jang [26]. Using a given input/output data set, the toolbox function ANFIS constructs a Fuzzy Inference System (FIS) whose membership function parameters are tuned (adjusted) using either a backpropagation algorithm alone or in combination with a least squares type of method. This adjustment allows your fuzzy systems to learn from the data they are modeling.

The modeling approach used by ANFIS is similar to many system identification techniques. First, you hypothesize a parameterized model structure (relating inputs to membership functions to rules to outputs to membership functions, and so on). Next, you collect input/output data in a form that will be usable by ANFIS for training. You can then use ANFIS to *train* the FIS model to emulate the training data presented to it by modifying the membership function parameters according to a chosen error criterion.

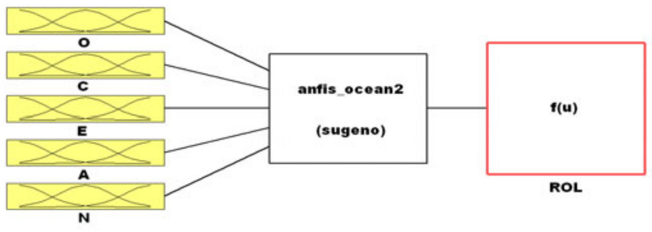
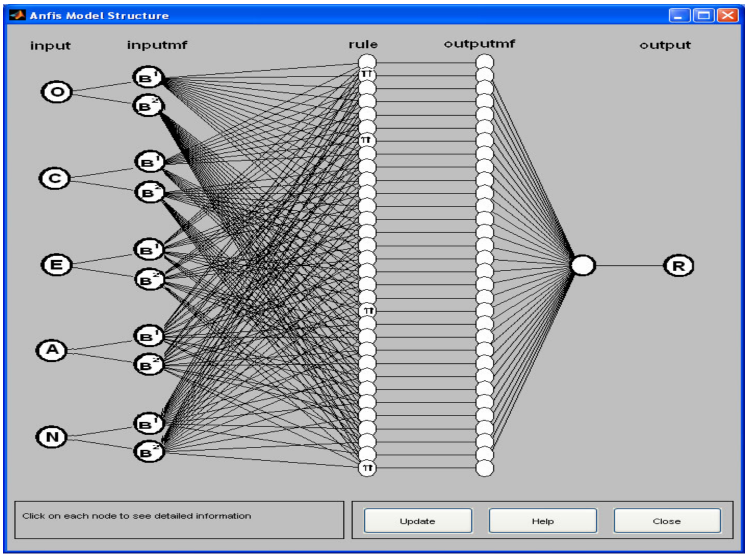
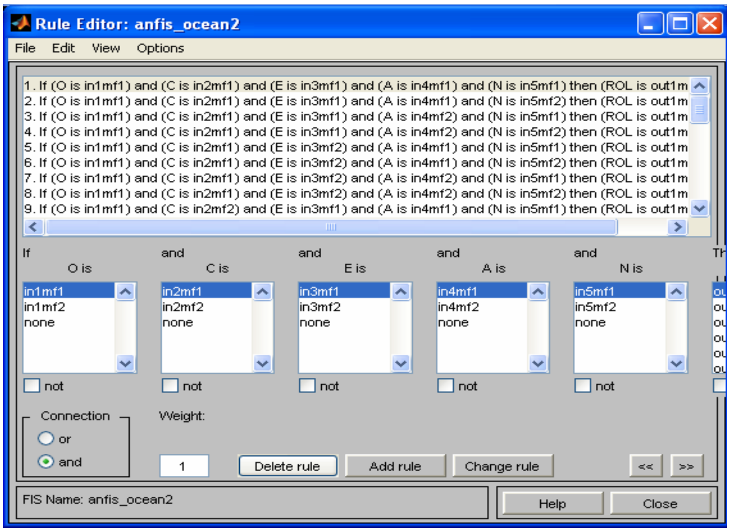


Fig. 1. ANFIS Big Five Test Model



Click on each node to see detailed information

Fig. 2. Big Five Test ANFIS Model Architecture



```
1. If (O is in1mf1) and (C is in2mf1) and (E is in3mf1) and (A is in4mf1) and (N is in5mf1) then (ROL is out1m
2. If (O is in1mf1) and (C is in2mf1) and (E is in3mf1) and (A is in4mf1) and (N is in5mf2) then (ROL is out1m
3. If (O is in1mf1) and (C is in2mf1) and (E is in3mf1) and (A is in4mf2) and (N is in5mf1) then (ROL is out1m
4. If (O is in1mf1) and (C is in2mf1) and (E is in3mf1) and (A is in4mf2) and (N is in5mf2) then (ROL is out1m
5. If (O is in1mf1) and (C is in2mf1) and (E is in3mf2) and (A is in4mf1) and (N is in5mf1) then (ROL is out1m
6. If (O is in1mf1) and (C is in2mf1) and (E is in3mf2) and (A is in4mf1) and (N is in5mf2) then (ROL is out1m
7. If (O is in1mf1) and (C is in2mf1) and (E is in3mf2) and (A is in4mf2) and (N is in5mf1) then (ROL is out1m
8. If (O is in1mf1) and (C is in2mf1) and (E is in3mf2) and (A is in4mf2) and (N is in5mf2) then (ROL is out1m
9. If (O is in1mf1) and (C is in2mf2) and (E is in3mf1) and (A is in4mf1) and (N is in5mf1) then (ROL is out1m
```

If O is	and C is	and E is	and A is	and N is
in1mf1	in2mf1	in3mf1	in4mf1	in5mf1
in1mf2	in2mf2	in3mf2	in4mf2	in5mf2
none	none	none	none	none
☐ not	☐ not	☐ not	☐ not	☐ not

Connection: ○ or [✓ ● and]

Weight: 1

FIS Name: anfis_ocean2

Fig. 3. Rules obtained in ANFIS Big Five Test Model

In general, this type of modeling works well if the training data presented to ANFIS for training (estimating) membership function parameters is fully representative of the features of the data that the trained FIS is intended to model.

Figure 1 shows our ANFIS model proposed where each Big Five trait is an input linguistic variable. Openness (O) takes a label value of one (1), conscientiousness (C) a label value of two (2), extroversion (E) a value of three (3), agreeableness (A) a value of four (4) and neuroticism (N) a value of (5). These input variables enter the ANFIS model and obtain an output variable that is the resulting Role recommended. Label values for Role are (1) analyst, (2) architect, (3) developer-programmer, (4) documenter, (5) tester and (6) presenter.

The ANFIS under consideration has five variable inputs denoted by $x = \{ O, C, E, A, N \}$, with two Gaussian membership functions (B), a set of 32 rules and one output variable Role (R). For a first-order Sugeno Fuzzy Model, a k-th rule can be expressed as:

Rule k-th:

IF (x_1 is B_1^k) AND (x_2 is B_2^k) AND (x_3 is B_3^k) AND (x_4 is B_4^k) AND (x_5 is B_5^k)

THEN R is $f^k(x)$, where

$$f^k = p_1^k + p_2^k + p_3^k + p_4^k + p_5^k + p_0^k$$

and membership functions are denoted by:

$$\mu_{B_i^k}(x_i) = \exp\left[-\tfrac{1}{2} \left(\frac{x_i - m_i^k}{\sigma_i^k} \right) \right]$$

where p_i^k are linear parameters, and B_i^k are Gaussian membership functions for k= 1,2,3, …, 32 and i = 1, …,5.

The corresponding equivalent ANFIS architecture is as shown in Fig. 2. The entire system architecture consists of five layers, these are {input, inputmf, rule, outputmf, output}.

Using this architecture, with OCEAN traits as input linguistic variables and Roles as output linguistic variables, a Takagi-Sugeno-Kang (TSK) FIS type model was developed using the ANFIS Editor GUI, creating, training and testing it, to find adequate rules to help us find best suited Role with OCEAN personality traits. Figure 3 is an abstract visualization of the rules generated by the model.

4 Results

Work of our case studies from 2007-2 to 2009-2, consisted of 72 software engineers working with real projects, assigning 80 roles in software development teams. Of these 13 have been assigned as Analysts, 13 Architects, 17 Developers/Programmers, 14 Documenters, 14 Tester and valuator, 9 Image and presentation.

Big Five personality test results obtained are presented in Table 1 showing the means and standard deviation of each trait for every role. Each table row is a personality vector unique for every Role, no two rows have exactly the same values for every one of their attributes, and this gives us a significant difference between each Role to recommend based on the Big Five Personality Test.

Table 1. Results of OCEAN test for Software Engineering Roles

	ROLE	O	C	E	A	N
		MEAN				
1	**Analyst**	**50.615**	**63.143**	**51.571**	**45.846**	**64.286**
2	**Architect**	**54.154**	**67.231**	**57.308**	**52.154**	**64.154**
3	**Developer**	**51.667**	**52.778**	**44.111**	**54.471**	**59.222**
4	**Documenter**	**55.000**	**61.286**	**51.929**	**52.000**	**58.857**
5	**Tester**	**51.714**	**66.133**	**56.000**	**47.857**	**63.333**
6	**Presenter**	**52.222**	**60.889**	**46.889**	**55.778**	**65.778**
		STANDARD DEVIATION				
1	**Analyst**	**11.644**	**9.502**	**19.394**	**12.688**	**11.472**
2	**Architect**	**8.735**	**10.910**	**16.317**	**10.846**	**7.978**
3	**Developer**	**16.439**	**10.429**	**12.718**	**11.737**	**11.944**
4	**Documenter**	**11.602**	**8.361**	**13.035**	**8.302**	**8.179**
5	**Tester**	**15.066**	**10.596**	**20.340**	**12.538**	**16.189**
6	**Presenter**	**15.279**	**15.136**	**12.534**	**14.507**	**9.871**
		TOTAL AVERAGE				
	Average	**52.625**	**61.125**	**50.513**	**51.250**	**62.525**

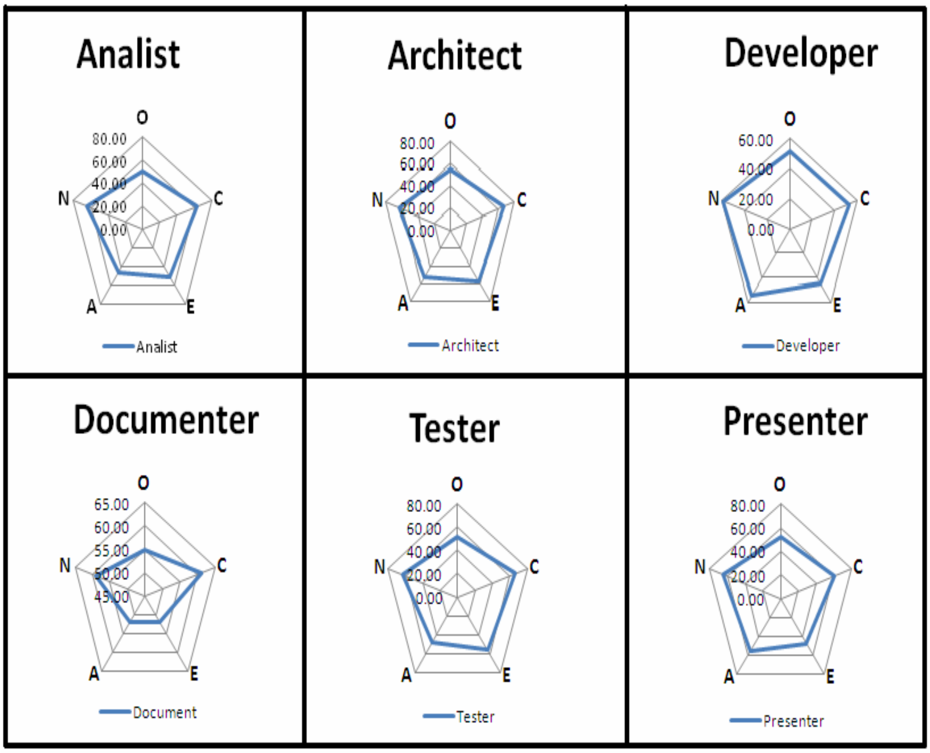


Fig. 4. Big Five Patterns (B5P)

To see the big picture these results can be displayed relating them with a center point using a radar chart type, obtaining Big Five Patterns (B5P) for Software Engineering Roles, these patterns are shown if figure 4, while figure 5 is a comparative of them.

There are significative differences between each role. For example trait (E) is high for architect and tester, low for a developer. Trait (C) is high for most roles except for developer. Trait (A) is high for a presenter and developer, but low for an analyst. This can give us a glimpse of specific traits for a particular role. Thus one trait does not define the personality of a role, but a personality vector with all traits involved can give us differences between each role.

These data was used for our ANFIS model obtaining figure 6 where it shows Input Trait and Output Role Relationships of this model. Whereas numeric values for linguistic variable Roles were (1) Analyst, (2) Architect, (3) Developer-Programmer, (4) Documenter, (5) Tester and (6) Presenter. Openness trait (O) covers roles 2 thru 4, Conscientiousness trait (C) engulfs roles 4 thru 6, Extroversion trait (E) from 3 thru 6, Agreeableness trait (A) reaches covers 2,3 and 4, and Neuroticism trait (N) reaches all roles.

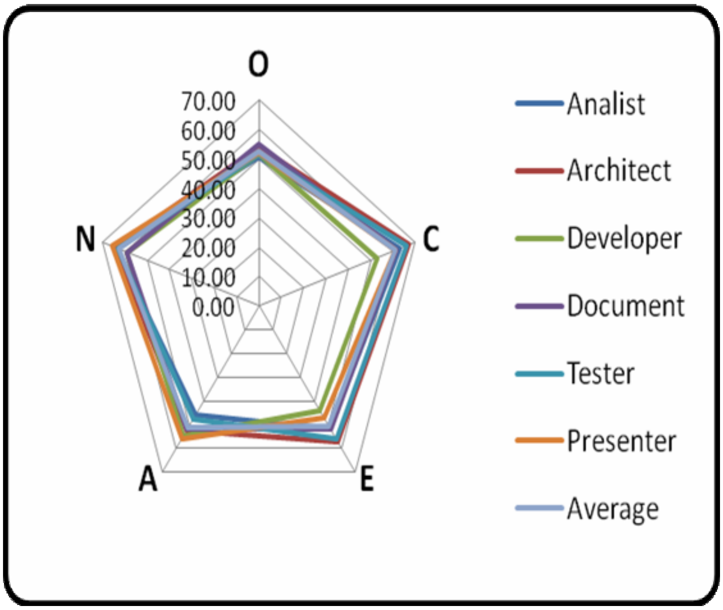


Fig. 5. Big Five Patterns Comparative

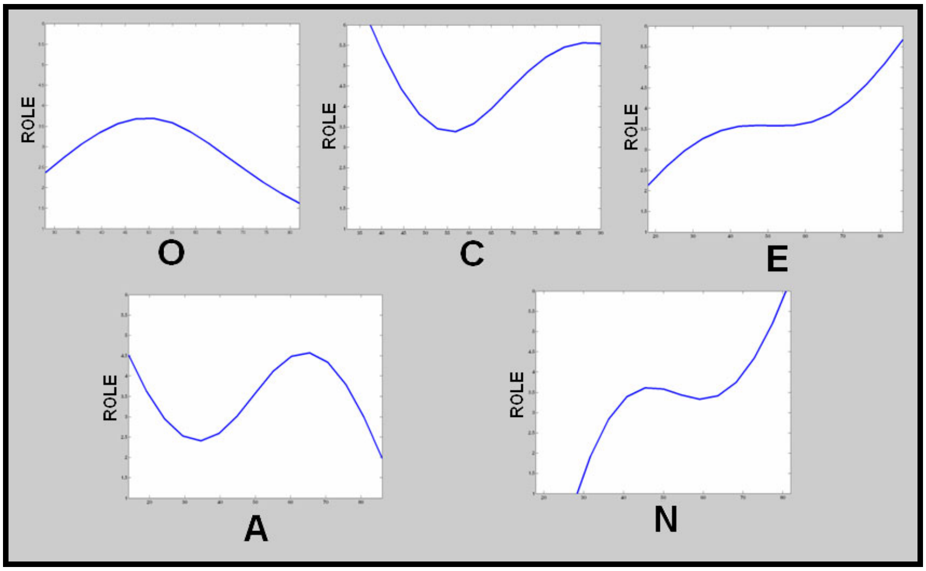


Fig. 6. Input Trait and Output Role Relationships

Analizing data, range of trait means are from 50 to 60, we will consider low degree around 30-40 and high degree around 70-80 based on standard deviation. With these results we can assertain that a low degree of (E) is definitively recomended to place this person as a Developer-Programmer, this indicates a person highly Introverted, for him is difficult to relate with others, although his high degree in trait (A) is an asset as he is very cooperative, trusting and compliant, attributes for a good programmer, also in B5P pattern figure of developer has a high degree of (O) indicating to be imaginative and creative, qualities for code design.

Trait (N) is a most significative trait as envelopes a wide range of roles, those with low degree of (N) or better said with opposite trait a high degree of (ES) Emotional Stability is a quality of a leader presenting security, reassurance and selfconfidence.

For a high degree of (E) as noted in B5P figures, an architect, analyst and tester present this quality, indicating that these roles are best suited for outgoing people, that can relate easilly with others, with passion and excitement in reaching goals and objectives.

The set of rules obtained with ANFIS learning approach implemented in MatLab's commercial Fuzzy Logic Toolbox [27], help us simulate our case studies and has given us another approach to start automating Role Assignment with RAMSET in software engineering projects.

5 Conclusions

The purpose of using RAMSET is identifying the individual's qualities to carry out the most suitable role in an effective working team. Some personalities and big five traits have been identified to better perform a type of role this defined by a personality vector. The Big Five personality test is only one of many personality tests applied and proposed in RAMSET, as we can see trait-role relationship is not clearly identifiable for all specific roles, a combination of these different personality tests will give us a better defined identifiable result.

Implementation of ANFIS models is a highly powerful tool to improve Data Base Rules arisen from this study; combination of different personality test FIS models will create a computer aided software tool invaluable for decision making in assignment of software engineering roles.

We know that personality is an important factor to performance of the team, thus is latent the difficulty to assign the adequate role to each member so the team can perform with success. Automation of RAMSET will help us with this task and follow up of RAMSET with fuzzy logic approaches will define better this tool and confirm RAMSET as a methodology for integrating teams and an instrument to select personnel for Software Engineering Development Teams.

References

1. Rothstein, M., Goffin, G.R.D.: The use of personality measures in personnel selection: What does current research support? Human Resource Management Review 16(2), 155–180 (2006)
2. Hough, L.M., Oswald, F.L.: Oswald: Personnel Selection: looking toward the future-remembering the past. Annual Reviews Psychology 51, 631–664 (2000)
3. Rodríguez, J.: Formación de grupos de desarrollo de software. Ediciones Yoltéotl, Guadalajara México (2004)

4. Mayer, D.B., Stalnaker, A.W.: Selection and evaluation of computer personnel – the research history of SIG/CPR. In: Proceedings of the 1968 23rd ACM National Conference, pp. 657–670 (1968)
5. Rutherfoord, R.H.: Using Personality Inventories to Help Form Teams for Software Engineering Class Projects. In: CITiCSE 2001, Canterbury UK, vol. 33(3), pp. 73–76. ACM, New York (2001)
6. Karn, J., Cowling, T.: A follow up study on the effect of personality on the performance of software engineering teams. In: ISESE 2006, Rio de Janeiro, Brazil, pp. 232–241 (2006)
7. Feldt, R., Torkar, R., Angelis, L., Samuelsson, M.: Towards Individualized Software Engineering: Empirical Studies Should Collect Psychometrics. In: CHASE 2008, Leipzig, Germany, May 13 (2008)
8. Gorla, N., Lam, Y.W.: Who Should Work With Whom? Building Effective Software Project Teams. Communications of the ACM 47(6), 79–82 (2004)
9. Capretz, L.F.: Personality types in software engineering. International Journal of Human-Computer Studies, 207–214 (2002)
10. Shen, S., Prior, S.D., White, A.S., Karamanoglu, M.: Using Personality Type Differences to Form Engineering Design Teams. Engineering Education 2(2), 54–66 (2007)
11. Wilde, D.J.: Creative teams, individual development and personality classification. ME310 Course Notes. Mechanical Engineering, Stanford University (2003)
12. Sodiya, A.S., Longe, H., Onashoga, S.A., Awodele, O.: An Improved Assessment of Personality Traits in Software Engineering. Interdisciplinary Journal of Information, Knowledge, and Management 2, 163–177 (2007)
13. Tupes, E.C., Cristal, R.E.: Recurrent Personality Factors Based on Trait Ratings. Technical Report ASD-TR-61-97, Personnel Laboratory, Air Force Systems Command, Lackland Air Force Base, TX (1961)
14. Digman, J.M.: Personality structure: Emergence of the five-factor model. Annual Review of Psychology 41, 417–440 (1990)
15. Goldberg, L.R.: The structure of phenotypic personality traits. American Psychologist 48, 26–34 (1993)
16. Costa Jr., P.T., McCrae, R.R.: Revised NEO Personality Inventory (NEO-PI-R) and NEO Five-Factor Inventory (NEO-FFI) manual. Psychological Assessment Resources, Odessa (1992)
17. Wei-Shen, T., Chung-Chian, H.: A Realistic Personnel Selection Tool Based on Fuzzy Data Mining Method. In: Proceedings of the 9th Joint Conference on Information Sciences, Kaohsiung, Taiwan (2006)
18. Werner, J.M.: Implications of AOCP and Contextual Performance for Human Resource Management. Human Resource Management Review 10(1), 3–24 (2000)
19. Tomayko, J.E.: Teaching a Project-Intensive Introduction to Software Engineering. SEI Carnegie Mellon University Tech. Rep., Pittsburgh Pennsylvania (1996)
20. Lather, A., Kumar S., Singh Y.: Suitability Assessment of Software Developers: A Fuzzy Approach. ACM SIGSOFT Software Engineering Notes 25(3) (May 30-31, 2000)
21. Oren, T.I., Ghasem-Aghaee, N.: Personality Representation Processable in Fuzzy Logic for Human Behavior Simulation. In: SCSC 2003, Montreal PQ, Canada July 20-24, pp. 11–18 (2003)
22. Oren, T.I., Ghasem-Aghaee, N.: Towards Fuzzy Agents with Dynamic Personality for Human Behavior Simulation. In: SCSC 2003, Montreal PQ, Canada July 20-24, pp. 3–10 (2003)

23. Jyothi, N.: Multi-View Technique for 3D Robotic Object Recognition System using Neuro-Fuzzy Method, `http://www.gisdevelopment.net/technology/ip/mi04039pf.htm`
24. Hui, H., Song, F.-J., Widjaja, J., Li, J.-H.: ANFIS-based fingerprint matching algorithm. Optical Engineering 43, 1814 (2004)
25. Gomathi, V., Ramar, K., Jeeyakumar, A.S.: Human Facial Expression Recognition Using MANFIS Model. International Journal of Computer Science and Engineering 3(2) (2009)
26. Jang, J.-S.R.: ANFIS: Adaptive Network Based Fuzzy Inference System. IEEE Transactions on Systems, Man, and Cybernetics 23(3) (May/June 1993)
27. Fuzzy Logic Toolbox: User's Guide of Matlab. The Mathworks, Inc. (1995-2009)

New Proposal for Eliminating Interferences in a Radar System

Carlos Campa*, Antonio Acevedo, and Elena Acevedo

Sección de Estudios de Posgrado, Escuela Superior de Ingeniería Mecánica y Eléctrica, Instituto Politécnico Nacional, Av. IPN s/n Col. Lindavista, 07738, Ciudad de México, México
ncra_58@hotmail.com, {macevedo,eacevedo}@ipn.mx

Abstract. In this work we present a new proposal to initialize the weights in a Backpropagation Neuronal Network (NN) using the coefficients from a FIR Low-Pass Filter to introduce a null in the radiation pattern in a seven-element array of antennas to eliminate interferences in a radar system. A radar system needs to eliminate the directional noise in order to obtain a cleaner signal. The method used to eliminate this kind of noise (jitter) has to be adaptive because the objective is in constant movement, therefore, the adaptation time must be as fast as possible. Our work is based on the window method to reduce the secondary lobes in fixed arrays of antennas. We modify the radiation pattern by introducing a null at 45.5° which corresponds to the secondary lobe where the interference is presented. This is achieved when we create windows from several FIR Low-Pass Filters. The coefficients of these filters are used to initialize the weight vectors of a Backpropagation Neural Network which performs the adaptive process to obtain the final parameters to achieve the noise elimination. For testing our proposal we calculate the Mean Square Error (MSE), the Signal Noise Relation (SNR) and we graphed the Radiation Pattern. In addition we calculated the Cross Correlation Index in each iteration, between the desired signal and our results. With this method we reduced the number of iterations required by the process.

Keywords: Radar system, Noise elimination, Backpropagation, FIR low-pass filter.

1 Introduction

In this paper we used an asymmetrical, distributed equidistant, seven-element array of antennas which was used by Widrow in 1967 [1], another work using this approach was proposed by Davisson [2]. We calculate the radiation pattern by summing the weighted outputs in each element. The weights are equal to one [3] in the array factor from a fixed array antenna. With these values, the filter impulse response from a rectangular window is formed and the magnitude response is a low-pass filter; therefore this window does not change the original radiation pattern.

First we analyze the fixed arrays of antennas to modify the radiation pattern. In particular, we describe the process for reducing the secondary lobes. This reduction is

* Corresponding author.

achieved by the use of Hamming, Hann and Kaiser windows, low-pass filter, and truncated low-pass filter with Blackman window. With the use of these filter coefficients, we could observe a side lobes reduction in the fixed antenna arrays.

It is important to emphasize the fact that with this method, the main lobe increased its half-power angle; this means that the main lobe became wider. The width of the beam is related to the cutoff frequency of filters which depends on the number of the coefficients. Then, there is no control on that frequency. Due to the above, in this work we proposed a truncated low-pass filter, since we need the cutoff frequency data to calculate the coefficients.

As a second step, the weights obtained by the windows are used to initialize the weight vector for the Backpropagation Neural Network used to implement the adaptive array of antennas. With this proposal we modified the original radiation pattern since the first iteration of the algorithm, and we achieved lower amplitude of side lobes with respect to the original pattern. Our goal is to reduce the amplitude of secondary lobes where the interferences are showed. Therefore, we need to place a null in the direction of one of the side lobes.

Finally, we modified the radiation pattern of the array by placing a null in the direction of an interfering signal known a priori. In this case the direction of the interference is located at 45.5 ° and 24°, where the secondary side lobe is. After training the Backpropagation neural network, we obtained the final weights for the adaptive array, considering that the main lobe must preserve the same half-power angle, and the null must be inserted in the specified direction.

For verifying the reduction in the number of iterations, we calculated the mean square error, the signal to noise ratio and then we plotted the radiation pattern to verify whether the null was inserted or not in the desired direction. In addition, for each step, we calculated the cross correlation index between the desired signal and the output of the array to determine the number of iterations required for each window to insert a null in the radiation pattern.

2 Materials and Methods

In this section we present the basic necessary concepts to understand our proposal. First, we present the antenna system used in this work. Then, we describe the secondary lobes reduction process. Finally, we present the design of the Backpropagation neural network.

2.1 Antenna System

The antenna system we used consists of an array of asymmetrical antennas with an equidistant separation between elements of $\lambda/2$ as it is shown in Figure 1, where λ is the wavelength. We can observe the generated radiation pattern, which consists of a main lobe at 0° and side lobes at ± 24 ° and ± 45.5 ° respecting to the vertical axis.

The side lobes or secondary lobes usually are unwanted because they radiate electromagnetic energy in one or more directions, so it is necessary to reduce or eliminate them. When Wu weights are equal either to one or to the coefficients of the windows, then we have fixed arrays [1] [2], and when the weights change over time it is an adaptive array.

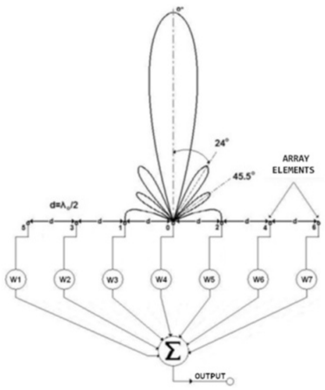


Fig. 1. Asymmetric array of dipole antennas

For obtaining the radiation pattern of the total far field antenna array, we used the superposition principle, which states that the field (E_θ) produced by a set of sources is the sum of the fields of individual sources of the Array Factor (AF) multiplied by the field element. To illustrate the radiation pattern of the asymmetric we obtained the array factor from Figure 1:

$$\begin{aligned} FA(\theta) = W_1 e^{(-6\mathrm{j}\pi \mathrm{d} \sin\theta)} e^{jw_0} + W_2 e^{(-4\mathrm{j}\pi \mathrm{d} \sin\theta)} e^{jw_0} \\ + W_3 e^{(-2\mathrm{j}\pi \mathrm{d} \sin\theta)} e^{jw_0} + W_4 e^{jw_0} + W_5 e^{(2\mathrm{j}\pi \mathrm{d} \sin\theta)} e^{jw_0} \\ + W_6 e^{(4\mathrm{j}\pi \mathrm{d} \sin\theta)} e^{jw_0} + W_7 e^{(6\mathrm{j}\pi \mathrm{d} \sin\theta)} e^{jw_0} \end{aligned} \tag{1}$$

In vectorial form

$$FA(\theta) = \mathrm{W}^{\mathrm{T}} \cdot a(\theta) \tag{2}$$

where:

$$\begin{aligned} \boldsymbol{a}(\theta) = e^{jw_0} * \big[e^{(-6\mathrm{j}\pi \mathrm{d} \sin\theta)}, \quad e^{(-4\mathrm{j}\pi \mathrm{d} \sin\theta)}, \quad e^{(-2\mathrm{j}\pi \mathrm{d} \sin\theta)}, \\ 1,\ e^{(2\mathrm{j}\pi \mathrm{d} \sin\theta)},\ e^{(4\mathrm{j}\pi \mathrm{d} \sin\theta)},\ e^{(6\mathrm{j}\pi \mathrm{d} \sin\theta)}\big] \end{aligned} \tag{3}$$

Then the field produced by the asymmetric array of antennas from Figure 1 is given by:

$$E_{\grave{e}} = \frac{jk\ \eta I_0 L e^{-jkr}}{4\eth r} \sin\grave{e} * FA \tag{4}$$

where L is the length of the dipole,η is the intrinsic impedance of the medium, θ is the angle respecting to the z-axis in spherical coordinates, and r is the distance from the antenna to a reference point. From the array factor in (1), the weight vector is taken as the impulse response of the rectangular window with 7 coefficients:

$$\boldsymbol{w}[n] = [W_1, W_2, W_3, W_4, W_5, W_6, W_7] = [1, 1, 1, 1, 1, 1, 1] \tag{5}$$

Fourier transform is applied to obtain the magnitude response of the filter, as follows:

$$\left|\boldsymbol{W}(e^{j\omega})\right| = \left|\sum_{n=1}^{7} \boldsymbol{w}[n]\, e^{-j\omega n}\right| \tag{6}$$

Figure 2 shows the magnitude response of the filter. We can observe a main lobe and several side lobes in the range from 0 to $\pi/2$. It shows a behavior of a low-pass filter with cutoff frequency of $\omega_c = 0.0640\pi$.

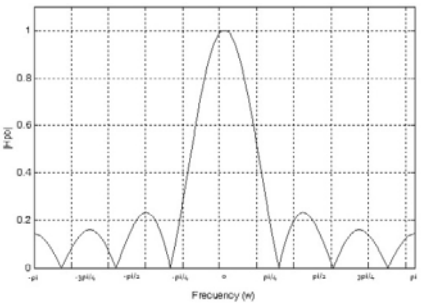


Fig. 2. Normalized magnitude response $\left|W\left(e^{j\omega}\right)\right|$

The value ω_c is used to calculate the low-pass filter coefficients which are used to initialize the weight vector of the Backpropagation neural network.

2.2 Secondary Lobes Reduction

In fixed arrays, the windows and low-pass filter coefficients are used in the factor array [2] [5] for reduction of side lobes. There are several window functions and methods to reduce the amplitude of secondary lobes. In this paper we present only some windows which are calculated by the following expressions:

$$Hamming(n) = 0.54 - 0.46\cos\left(2\pi\frac{n}{N}\right),\ 0 \le n \le N \tag{7}$$

$$Hann(n) = 0.5\left(1 - \cos\left(2\pi\frac{n}{N}\right)\right),\ 0 \le n \le N \tag{8}$$

$$Kaiser(n) = \begin{cases} \dfrac{I_0\left(\pi\alpha\sqrt{1-(2k/n-1)^2}\right)}{I_0(\pi\alpha)} & 0 \le k \le n \\ 0 & \text{other case} \end{cases} \tag{9}$$

where: $I_0(X)$ is the Bessel function.

$$Blac(n) = (-1)^n \cos\left(\frac{2\pi n}{N-1}\right) \tag{10}$$

We can notice that ω_c is not needed to obtain the coefficients. The number of coefficients, n, varies depending on the number of elements in the antenna array.

From figure 1 we concluded that seven coefficients ($n = 7$) are needed. Now, taking as a reference the impulse response of the rectangular window in (5) and knowing that this window behaves as a low-pass filter, we proposed to calculate the coefficients for a low-pass FIR filter truncated, because in contradiction to the window functions, the value of the cutoff frequency is taken into account and we can control the width of the main lobe. The frequency response of an ideal low-pass filter has a linear phase response. The truncated filter takes only some values of the coefficients of the impulse response in the range of $-M \leq n \leq M$ and the coefficients out of the range are equal to zero, therefore we have a finite length of $N = 2M + 1$:

$$h_{Pb}[n] = \begin{cases} \dfrac{\sin \omega_c(n-M)}{\pi(n-M)} , & 0 \leq n \leq N-1 \\ 0 & Other\ case \end{cases} \tag{11}$$

The coefficients obtained in (7 – 11) are used to initialize the weight vector in the adaptive antenna array. The goal of this proposal is to reduce the number of iterations taken by the adaptation process to insert a null. In this paper, the direction of the interference coincides with the angle of the side lobes at 24º, 45.5º and 90º.

2.3 Backpropagation Neural Network

The Backpropagation model proposed by Rumelhart and McClelland has become one of the most powerful tools for pattern recognition in the neural network approach. It reduces the mean quadratic error between the desired signal and the current output of the network with respect to the connection weights in each iteration. The structure of a multilayer neural network with one hidden layer was used in this work Figure 3:

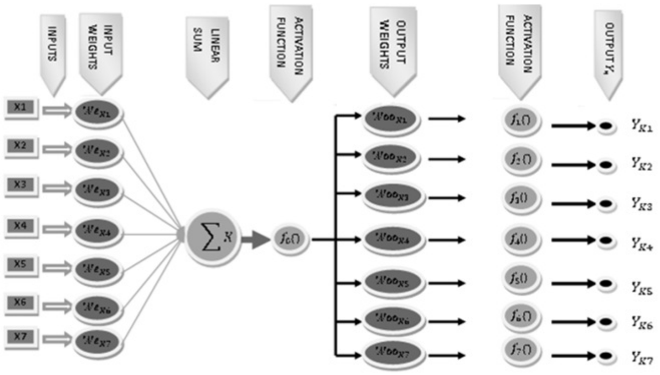


Fig. 3. Backpropagation neural network structure

The input and the output layer have seven neurons which correspond to the seven-element antenna array. We used the nonlinear sigmoid activation function.

Table 1 shows the coefficients values used to initialize the weight vector of the Backpropagation neural network. These values were obtained after applying the different windows.

Table 1. Coefficients used for weight vector initialization for Backpropagation neural network

N	W_{Ones}	$W_{WHamming}$	W_{WHann}	$W_{WKaiser}$	W_{FPBN}	$W_{FPBBlack}$
1	1	0.0800	0	0.9403	0.9405	0
2	1	0.3100	0.2500	0.9732	0.9733	0.0083
3	1	0.7700	0.7500	0.9933	0.9933	0.0403
4	1	1.0000	1.0000	1.0000	1.0000	0.0630
5	1	0.7700	0.7500	0.9933	0.9933	0.0386
6	1	0.3100	0.2500	0.9732	0.9733	0.0077
7	1	0.0800	0	0.9403	0.9405	0

The low-pass filter coefficients from FPB window were normalized because they yielded the best results. The neural network places a zero at a time and calculates the Signal to Noise Ratio (SNR), the Root Mean Square Error (MSE) and the correlation index between the desired signal and the output.

Equation (12) is used to calculate the signal to noise ratio (SNR), Mean Square Error (MSE) is calculated with (13-15) and (16) is used for calculating the maximum rate correlation (MC) of the output signal.

$$SNR = 10\ log_{10} \frac{(We \times \overline{vS}) \cdot (vS \times \overline{We})}{(We \times \overline{vI}) \cdot (vI \times \overline{We})} \tag{12}$$

where:

- vS = desired signal vector
- vl = interference angle vector
- We = input weights

$$MSE = \sum_{x=1}^{n} ep^2 \tag{13}$$

$$ep = {}^1/_2 \sum_{k=1}^{s} (\delta_k)^2 \tag{14}$$

$$\delta_k^s = (t_k - y_k^s) \tag{15}$$

where:

- ep = induced error for each pattern
- δ_k^s = error in each output

- t_k = desired signal
- y_k^s = output network in each neuron

$$MC = max\left(\frac{\sum_{n=-\infty}^{\infty} S_D[n] * S_{sal}[n-k]}{\sqrt{(\sum_{n=-\infty}^{\infty} S_D[n] * S_D[n-k]) * (\sum_{n=-\infty}^{\infty} S_{sal}[n] * S_{sal}[n-k])}}\right) \tag{16}$$

where:

- S_{sal} = output signal
- S_D = desired signal

The resulting parameters from the Backpropagation neural network introduced a zero in the side lobes at ± 45.5° ± 24°.

3 Experiments and Results

For testing the results from each window we calculated the following parameters from iterations 6 through 9:

1.- Signal to noise ratio.
2.- Root mean square error.
3.- Correlation index.

Table 2 shows the case when a zero is introduced in the first secondary lobe at 45.5°. We can observe that Kaiser window presents a signal to noise ratio of 69.210041 dB which is the greatest value in iteration 9. Result from FPBN window is very similar to Kaiser window with a value of 69.192511 dB. On the other hand, Kaiser induces a very small MSE of 0.445892 which is slightly greater than the obtained by the windows: Ones equal to 0.413471 and FPBN equal to 0.445830. Finally, the correlation index between the desired signal and filtered signal is 93.718763 for Kaiser, 93.713275 for FPBN, and 91.867102 for Ones windows.

There are correlation index values greater than the above results such as Hamming and Hann windows with 99.967892 and 99.959186, respectively; however, Kaiser, Ones and FPBN windows provided better results for the three benchmarks.

Now, we need to plot the radiation pattern to find out which window introduces the null in the desired direction.

Observing figures 5, 6 and 9 we notice that Hamming, Hann and FPB with Blackman window affect the half-power angle because they increase it. This implies that the gain or the directivity of the main lobe decrease just as the efficiency, and what we look for is to concentrate the greatest quantity of energy in the main lobe with the lowest half-power angle. However, Kaiser window (figure 7) shows the best results because the secondary lobe at 45.5 ° is almost vanished in 7 iterations (blue-circled line).

Figure 4 shows the radiation pattern when Ones window is used for initialization. It can be observed that the shape of the main lobe is preserved at any iteration, and in 8 and 9 iterations, the Ones window introduces a null in the assigned direction. According to Table 2, this window shows the greatest SNR and the lesser MSE, although the correlation index presents low percentage.

Table 2. The values of the parameters SNR, MSE and CI from iterations 6 through 9 when a zero is introduced at 45.5º

Window	Number of Iterations	SNR (dB)	MSE	Correlation Index (%)
Ones	6	31.212157	1.266932	81.710410
	7	40.915319	0.647964	85.987729
	8	52.387430	0.439700	89.364769
	9	64.169735	0.413471	91.867102
Hamming	6	53.120361	2.618072	99.903037
	7	55.881995	2.552359	99.935632
	8	58.643200	2.489031	99.955330
	9	61.387413	2.427984	99.967892
Hann	6	51.925578	3.029346	99.874932
	7	54.781435	2.949531	99.917795
	8	57.758855	2.872518	99.943189
	9	60.899062	2.798211	99.959186
Kaiser	6	37.526094	0.778117	85.349271
	7	47.750276	0.501622	88.999118
	8	58.422886	0.453040	91.735617
	9	69.210041	0.445892	93.718763
FPBN	6	37.506085	0.779135	85.338363
	7	47.729840	0.501797	88.990206
	8	58.403802	0.452999	91.728601
	9	69.192511	0.445830	93.713275
FPB with Blackman window	6	19.266619	6.859073	89.040618
	7	18.914237	6.828505	92.341726
	8	18.617182	6.792110	94.568791
	9	18.366871	6.749118	95.427260

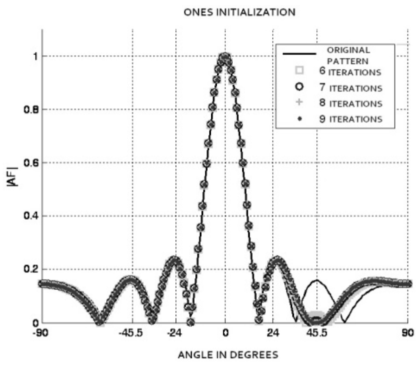


Fig. 4. Ones window initialization

Figure 5 illustrates the radiation pattern when initializing with the Hamming window. We can observe that the main lobe is wider than the original. However, a null is inserted in the assigned direction in each iteration. According to Table 2, this window provides the best correlation index between the desired signal and the filtered signal, because it inserts a zero very quickly, but does not preserve the half-power angle of the original pattern.

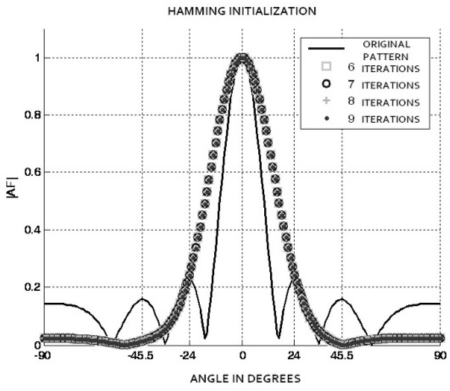


Fig. 5. Hamming window initialization

Figure 6 shows the radiation pattern when the Hann window is used for initializing. We can notice that the main lobe is wider than the original and is even wider than the obtained with the Hamming window. Like Hamming, it inserts a null in the assigned direction in each iteration and provides a great correlation index between the desired signal and the filtered signal. It inserts a zero very quickly, but does not preserve the half-power angle of the original pattern.

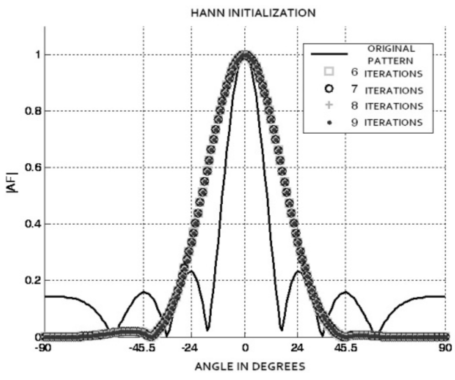


Fig. 6. Hann window initialization

Figure 7 shows the radiation pattern with the Kaiser window initialization. The shape of the main lobe is preserved in any iteration. Null is introduced in the assigned directions until iterations 8 and 9 are reached. We can observe from Table 2 that SNR is one of the largest and presents a low MSE, although the correlation index percentage is low compared with Hann and Hamming window results.

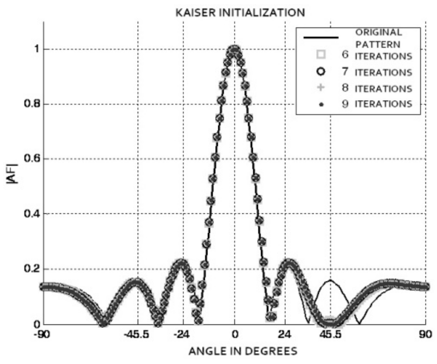


Fig. 7. Kaiser window initialization

Figure 8 shows the radiation pattern when we use the FPBN window initialization. This window presents a similar behavior to Ones and Kaiser windows. According to Table 2 the SNR is one of the largest and has a small MSE, although the rate of correlation has a low percentage compared with Hann and Hamming window.

Same process is performed to find out which windows provides the coefficients for initializing the weight vector of Backpropagation neural network that will produced best results to eliminate the secondary lobe at 24°, but discarding the three above windows which widened the main lobe.

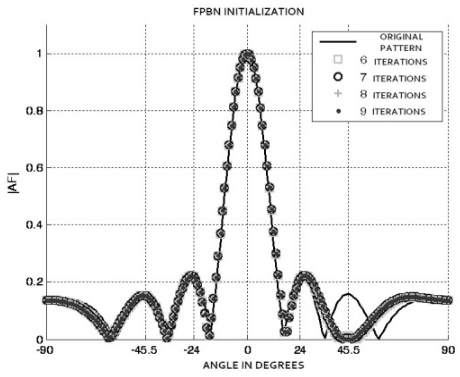


Fig. 8. FPBN window initialization

Table 3 shows the benchmarks when a zero is introduced at 24°. It can be observed that more iterations are required to achieve the goal. This is because the closer the secondary lobe is from the main lobe the harder is the elimination of the lobe because it contains more energy. The results show that the FPBN window presents the best efficiency, at iteration 12, it shows a SNR equal to 147.697961dB and the lowest MSE equal to 0.479700 but the best correlation index is obtained with the Kaiser window (94.683943%).

Table 3. The values of the parameters SNR, MSE and CI from iterations 9 through 12 when a zero is introduced at 24°

Window	**Number of Iterations**	**SNR (dB)**	**MSE**	**Correlation Index (%)**
Ones	9	20.373207	2.869320	92.276890
	10	25.091940	2.206256	93.102566
	11	36.996036	1.223430	94.064350
	12	69.032278	0.520028	94.473918
Kaiser	9	39.328505	1.075121	91.497611
	10	84.732492	0.519505	92.649002
	11	117.221341	0.484442	93.732544
	12	147.472620	0.479735	94.683943
FPBN	9	39.245646	1.078720	91.496478
	10	84.987599	0.520054	92.646636
	11	117.464164	0.484406	93.729652
	12	147.697961	0.479700	94.680936

Figure 9 shows the radiation pattern with the Ones window initialization. The shape of the main lobe is preserved and the zero is introduced in the assigned direction at iteration 12. According to Table 3, this window presents the worst results with a SNR = 69.032278 dB, a MSE = 0.520028 and CI = 94.473918.

We can observe the results from the Kaiser window initialization in figure 10. The shape of the main lobe is preserved. At iterations 10 and 11 the null has been inserted at the assigned direction. According to Table 3, its SNR is one of the largest, it presents a low MSE and the greatest percentage of correlation index.

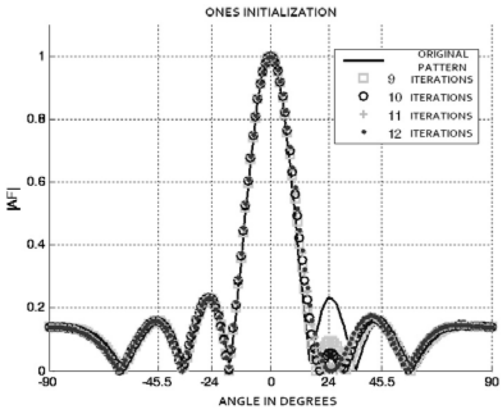


Fig. 9. Ones window initialization

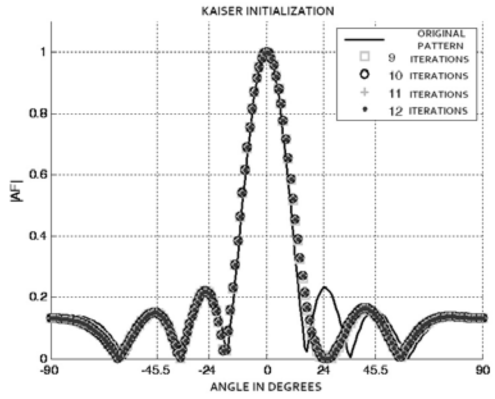


Fig. 10. Kaiser window initialization

Figure 11 shows the radiation pattern when the FPBN window is used for initialization. According to Table 3, this window presents the largest SNR, the lowest MSE and the correlation index is slightly lower than the obtained by the Ones window.

From the above results, we can conclude that the FPBN window is the best option for initializing the weight vector for Backpropagation neural network because it obtains the suitable results at iteration 10.

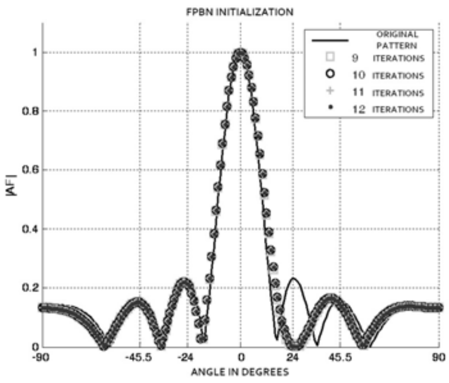


Fig. 11. FPBN window initialization

Table 4 shows the best results obtained by FBPN and Kaiser windows for eliminating secondary lobes at 24º and 45.5º, respectively.

Table 4. The best results obtained by FBPN and Kaiser windows to eliminate the secondary lobes at 24º and 45.5º, respectively

Angle	Window	Iteration	SNR (dB)	MSE	Correlation Index (%)
24º	**FPBN**	**10**	84.987599	0.520054	92.646636
45.5º	**Kaiser**	**7**	47.750276	0.501622	88.999118

4 Conclusions

The behavior of several windows is similar to a low-pass FIR filter, therefore their results can be considered as filter coefficients.

When we applied these values to initialize the weight vector of a Backpropagation neural network, we could observe a reduction in the number of iterations required for eliminating secondary lobes in a radiation pattern from a seven-element antenna array.

Some windows such as Ones, Hamming, Hann, Kaiser, FBPN and FBPN with Blackman were used to achieve the elimination of secondary lobes at 24º and 45.5º, where an *apriori* known interference is located.

From the obtained results, we could observe that Hamming, Hann and FBPN with Blackman window widened the main lobe therefore they were discarded.

Two windows were suitable for providing the coefficients to initialize the weight vector of the Backpropagation neural network and for eliminating secondary lobes: FBPN window to introduce a null at 24º and Kaiser window to introduce a null at 45.5º.

The use of several windows affected the number of iterations required to achieve the secondary lobes elimination. In this work, the best results were reached at iteration 10 and 7, when we applied FBPN and Kaiser windows, respectively.

In other studies we can observe the reduction of the secondary lobes, but that solution needs a lot of iterations. They proposed a solution to obtain the weights but they do not consider the processing time as an issue because they applied it in fixed antenna arrays [5].

Acknowledgments. The authors would like to thank the Instituto Politécnico Nacional (COFAA and SIP), and SNI for their economical support to develop this work.

References

1. Widrow, B., Mantey, P.E., Griffiths, L.J., Goode, B.B.: Adaptive Antenna Systems. Proceedings of the IEEE (1967)
2. Davisson, L.D.: A theory of adaptive filtering. IEEE Trans. Information Theory 12, 97–102 (1966)
3. Gross, F.B.: Smart Antennas for Wireless Communications. McGraw-Hill, New York (1995)
4. Sung, S., Ham, F.M., Shelton, W.: A new robust neuronal network method for coherent interference rejection in adaptive array systems. In: Lu, M., He, Z. (eds.) IJCNN International Joint Conference on Neural Networks 1990. IEEE Transactions Antennas and Propagation, vol. 2, pp. 119–124 (1993)
5. Yu, C.S., Mitra, S.K.: Digital Signal Processing, 2nd edn. McGraw-Hill, New York (2001)
6. Brookner, E.: Trends in Array Radars for the 1980s and Beyond. IEEE Antenna and Propagation Society Newsletter (April 1984)
7. Steyskal, H.: Digital Beamforming Antennas—An Introduction. Microwave Journal, 107–124 (January 1987)
8. Liberti, J., Rappaport, T.: Smart Antennas for Wireless Communications: IS-95 and Third Generation CDMA Applications. Prentice Hall, New York (1999)
9. Margerum, D.: Self-Phased Arrays. In: Microwave Scanning Antennas, vol. 3, Array Antennas
10. Special Issue on Active and Adaptive Antennas IEEE Trans. Antennas and Propagation AP-12 (March 1964)

Type-2 Fuzzy Inference System Optimization Based on the Uncertainty of Membership Functions Applied to Benchmark Problems

Denisse Hidalgo[1], Patricia Melin[2], and Oscar Castillo[2]

[1] School of Engineering UABC University,
Tijuana, México
[2] Division of Graduate Studies Tijuana Institute of Technology,
Tijuana, México
paulette1019@hotmail.com, epmelin@hafsamx.org,
ocastillo@tectijuana.mx

Abstract. In this paper we describe a method for the optimization of type-2 fuzzy systems based on the level of uncertainty considering three different cases to reduce the complexity problem of searching the solution space. The proposed method produces the best fuzzy inference systems for particular applications based on a genetic algorithm. We apply a Genetic Algorithm to find the optimal type-2 fuzzy system dividing the search space in three subspaces. We show the comparative results obtained for the benchmark problems.

Keywords: Type-2 Fuzzy Logic, Genetic Algorithms.

1 Introduction

We describe in this paper a method for the optimization of type-2 fuzzy systems. The main goal of this research was to develop the optimization method for type-2 fuzzy systems based on the footprint of uncertainty (FOU) of the membership functions; we use benchmark problems to test the optimization method for the fuzzy systems and describe the simulation results.

The motivation for this work is that currently there are no systematic methods for constructing optimal type-2 fuzzy systems for particular applications. In most of the cases an optimization method is used, but no clear explanation of the solution is obtained. For these reasons we propose a systematic method for optimizing type-2 fuzzy systems based on the FOU.

This paper is organized as follows: Section 2 shows an introduction to the theory of soft computing techniques, section 3 describes the development of the evolutionary method; in section 4 the simulation results are presented, section 5 shows the conclusions and finally the references are presented.

2 Soft Computing Techniques

The term soft computing refers to a family of computing techniques comprising four different areas: Fuzzy Logic (FL), evolutionary computation (EC), neural networks (NN), and probabilistic reasoning (PR). The term soft computing distinguish these techniques from hard computing that is considered less flexible and computationally demanding. Imprecision results from our limited capability to resolve detail and encompasses the notions of partial, vague, noisy and incomplete information about the real world [1, 2, 3, 4, 5].

2.1 Type-2 Fuzzy Logic

The original theory of Fuzzy logic (FL) was proposed by Lotfi Zadeh [6], more than 40 years ago, and this theory cannot fully handle all the uncertainty that is present in real-world problems. Type-2 Fuzzy Logic can handle uncertainty because it can model and reduce it to the minimum their effects. Also, if all the uncertainties disappear, type-2 fuzzy logic reduces to type-1 fuzzy logic, in the same way that, if the randomness disappears, the probability is reduced to the determinism [4, 7, 8].

Fuzzy sets and fuzzy logic are the foundation of fuzzy systems, and have been developed looking to model the form that the brain manipulates inexact information. Type-2 fuzzy sets are used to model uncertainty and imprecision; originally they were proposed by Zadeh in 1975 and they are essentially "fuzzy-fuzzy" sets in which the membership degrees are type-1 fuzzy sets (See Fig. 1) [4, 5, 9, 10, 11, 12, 13, 16].

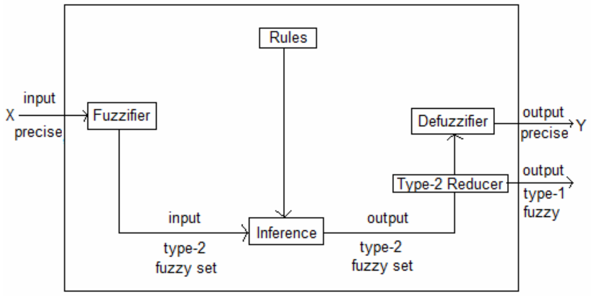


Fig. 1. Basic structure of Type-2 Fuzzy Inference System

2.2 Genetic Algorithms

To use a genetic algorithm (GA) one should represent a solution to the problem as a *genome* (or *chromosome*). The genetic algorithm then creates a population of solutions and applies genetic operators such as mutation and crossover to evolve the solutions in order to find the best one. It uses various selection criteria so that it picks the best individuals for mating (and subsequent crossover).

The objective function determines the best individual where each individual must represent a complete solution to the problem you are trying to optimize. Therefore the three most important aspects of using genetic algorithms are:

(1) definition of the objective function,
(2) definition and implementation of the genetic representation, and
(3) definition and implementation of the genetic operators [14].

3 Method Description

Based on the theory described above, a new method for optimization of the membership functions of type-2 fuzzy systems based on the level of uncertainty is proposed. In particular, the method includes optimizing the membership functions based on the uncertainty of the type-2 fuzzy system.

3.1 Optimization of the Fuzzy System Based on the Level of Uncertainty

Fuzzy systems can be robust and flexible in domains subject to imprecision and uncertainty. The linguistic representation of knowledge allows a human to interact with a fuzzy system in an intuitive, seamless manner [5].

For the Fuzzy Inference System optimization based on the level of uncertainty; the first step is obtain the optimal type-1 Fuzzy Inference System, which allows us to find the uncertainty of their membership functions. In this case we used Ɛ (epsilon) for representing the uncertainty. For the next step we used three cases to manage the use of genetic algorithms for all situations. These are:

1. Equal value of uncertainty for all membership function. We can see in Fig. 2 different membership functions with the same footprint of uncertainty.

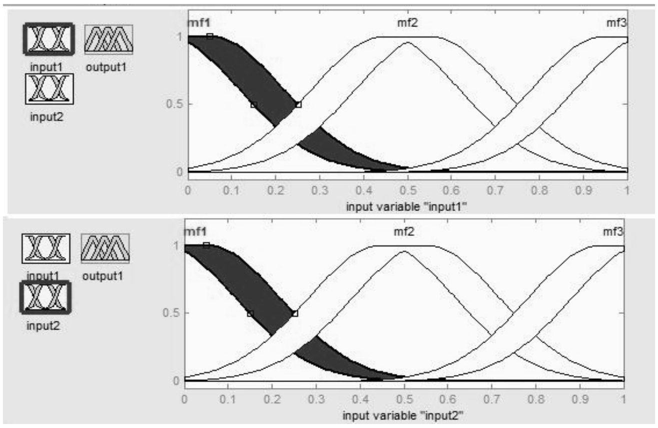


Fig. 2. Representation of equal value of uncertainty for all membership function

2. Different value of uncertainty in each input. We can see in Fig. 3 different membership functions with the same footprint of uncertainty for each input in the fuzzy system.

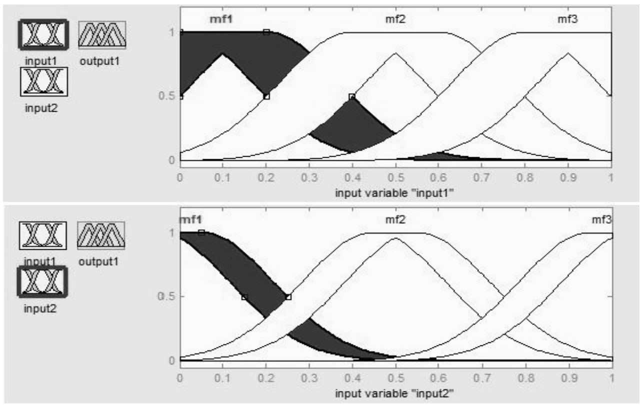


Fig. 3. Representation of different value of uncertainty in each input

3. Different value of uncertainty for each membership function. We can see in Fig. 4 different membership functions with different footprint of uncertainty.

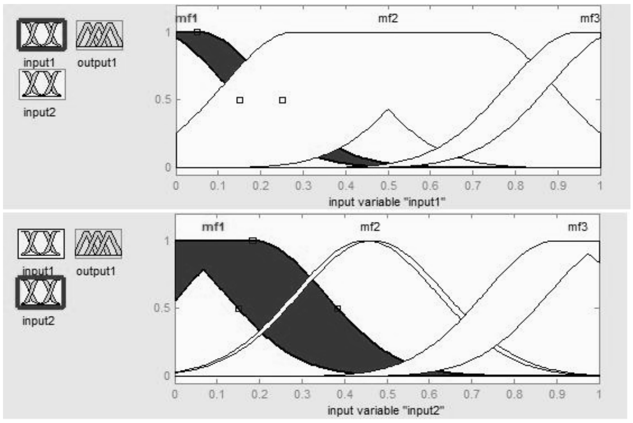


Fig. 4. Representation of different value of uncertainty for each membership function

In Fig. 5 we can see how the inputs in fuzzy inference systems affect the output, to find the largest percentage of data inside the output interval.

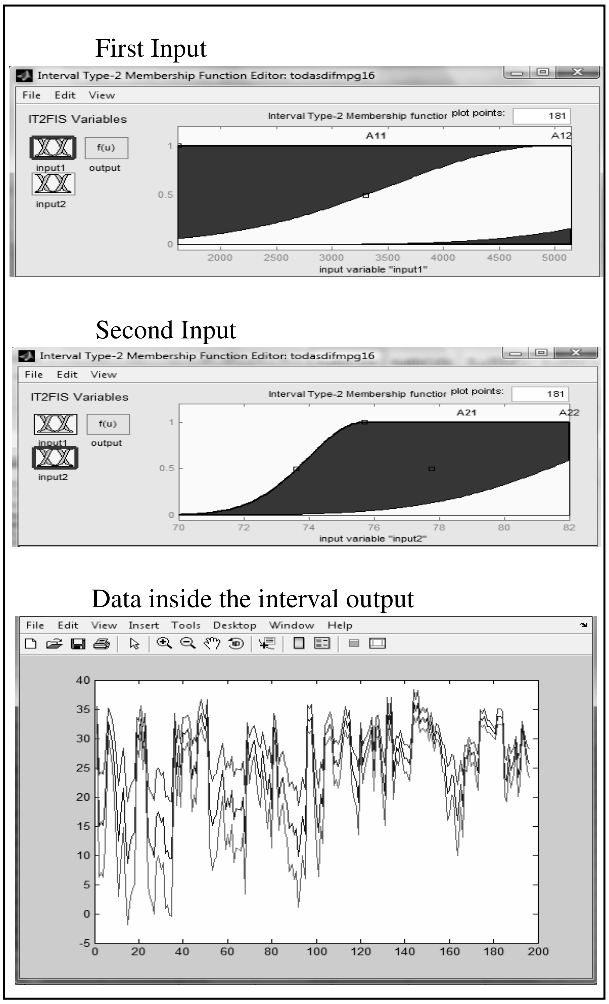

First Input
Interval Type-2 Membership Function Editor: todasdifmpg16
File Edit View
IT2FIS Variables
input1
output
input2
f(u)
Interval Type-2 Membership functior plot points: 181
A11
A12
input variable "input1"
Second Input
Interval Type-2 Membership Function Editor: todasdifmpg16
A21
A22
input variable "input2"
Data inside the interval output
File Edit View Insert Tools Desktop Window Help

Fig. 5. Data inside the output interval, where superior line indicates the upper value in the interval output, inferior line indicates the lower value in the interval output and middle line indicates the original output value of the fuzzy inference system

We consider the optimization of type-2 fuzzy systems based on the above mentioned three cases. As a consequence our proposed method is illustrated in Fig. 6.

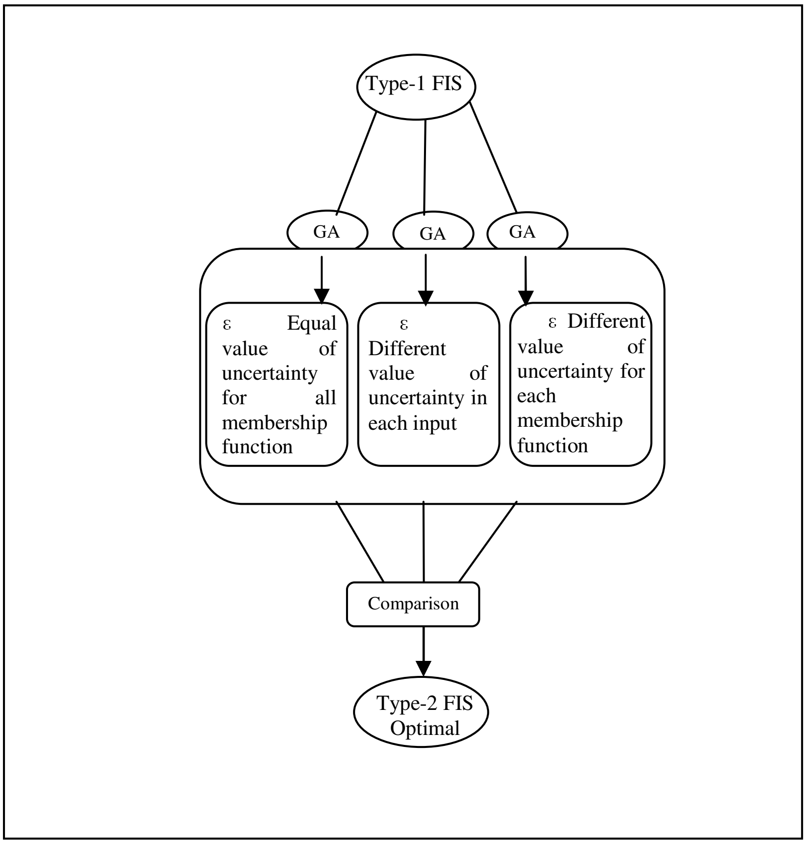

Type-1 FIS
GA
GA
GA
ε Equal value of uncertainty for all membership function
ε Different value of uncertainty in each input
ε Different value of uncertainty for each membership function
Comparison
Type-2 FIS Optimal

Fig. 6. Representation of Optimization to Type-2 Fuzzy Inference Systems based on level of uncertainty

This method focuses on the optimization of type-2 fuzzy systems with respect to the footprint of uncertainty of their membership functions; the main thing is to get a type-2 fuzzy system that is optimal for the problem to be solved; making a comparison between the performance of the three previous cases where we use a genetic algorithm to obtain the type-2 fuzzy logic system, penalized by the width of the footprint of uncertainty.

4 Simulation Results

We describe in this section simulation results for two benchmark problems. The type-2 fuzzy systems were implemented with the type-2 fuzzy logic toolbox developed previously by our group [10].

4.1 Miles Per Gallon (MPG) Benchmark Problem

The first benchmark problem is known as automobile MPG (Miles per Gallon) prediction. This problem is a typical nonlinear regression problem, in which several attributes (inputs variables) as number of cylinders, weight, model years, etc., are used to predict another continuous attribute (output variable) in this case MPG. The original data base contains 384 data, the first 192 were used for training data and the other 192 for testing. Jang et al. in [15], found the best model takes "weight" and "model year" as the inputs variables for the type-1 fuzzy inference system with two membership functions for each input.

Based on those results to obtain a type-2 fuzzy inference system we use the type-1 fuzzy system as a basis to manually perturb its value with an epsilon in the membership functions, in this case of the two most significant variables and reducing the input data for training, the following results are obtained. We propose an index to evaluate the fitness of the fuzzy systems:

$$Index = \frac{N}{1 + n\varepsilon}$$

Where N equals to data inside the interval output between the total output data; n equals to number of fuzzy systems; and ε equals to value of increased uncertainty of the membership functions for table 1, 2 and 3.

For the first case, 20 fuzzy systems were obtained by increasing the values of equal epsilon by 10%. For case two, where the increase of epsilon is different per input, 30% increase for the first input and 10% in second input; and for the third case where the increase of epsilon is different per membership function 15% 5% 10% 20% increase respectively.

Table 1 shows the simulation results with manual increase for Equal Value of Uncertainty for all Membership functions, for the MPG benchmark problem.

Table 1. Equal value of uncertainty with manually increase (10% Increase)

No.	**N**	**n**	**Epsilon (ε)**	**Index**	**Data inside the interval output**
Base	0.3418	20	0	0.11	67/192
1	0.602	20	0.1	0.2	118/192
2	0.7296	20	0.2	0.24	143/192
3	0.7857	20	0.3	0.26	154/192
4	0.7908	20	0.4	0.26	155/192
5	0.8061	20	0.5	0.27	158/192
6	0.8265	20	0.6	0.28	162/192
7	0.8469	20	0.7	0.28	166/192
8	0.8571	20	0.8	0.29	168/192
9	0.8571	20	0.9	0.29	168/192
10	0.8622	20	1.0	0.29	169/192
11	0.8622	20	1.1	0.29	169/192
12	0.8622	20	1.2	0.29	169/192
13	0.8622	20	1.3	0.29	169/192
14	0.8622	20	1.4	0.29	169/192
15	0.8622	20	1.5	0.29	169/192
16	0.8622	20	1.6	0.29	169/192
17	0.8622	20	1.7	0.29	169/192
18	0.8622	20	1.8	0.29	169/192
19	**0.8673**	**20**	**1.9**	**0.29**	**170/192**
20	0.8673	20	2.0	0.29	170/192

Table 2. Different value of uncertainty in each input with manually increase (30% Increase first input and 10% in second input)

No.	N	n	Epsilon (ε)	Index	Data inside the interval output
Base	0.3418	20	0	0.07	67/192
1	0.6531	20	0.2	0.13	128/192
2	0.7602	20	0.4	0.15	149/192
3	0.8214	20	0.6	0.16	161/192
4	0.852	20	0.8	0.17	167/192
5	0.8571	20	1.0	0.17	168/192
6	0.8622	20	1.2	0.17	169/192
7	0.8622	20	1.4	0.17	169/192
8	0.8622	20	1.6	0.17	169/192
9	0.8622	20	1.8	0.17	169/192
10	0.8673	20	2.0	0.17	170/192
11	0.8827	20	2.2	0.18	173/192
12	**0.8929**	**20**	**2.4**	**0.18**	**175/192**
13	0.8827	20	2.6	0.18	173/192
14	0.8622	20	2.8	0.17	169/192
15	0.8469	20	3.0	0.17	166/192
16	0.8469	20	3.2	0.17	166/192
17	0.8469	20	3.4	0.17	166/192
18	0.8469	20	3.6	0.17	166/192
19	0.8469	20	3.8	0.17	166/192
20	0.8469	20	4.0	0.17	166/192

Table 2, shows the simulation results with manual increase for different value of uncertainty in each input for the benchmark problem.

Table 3, shows the simulation results with manual increase for different value of uncertainty in each Membership function (15% 5% 10% 20% Increase respectively) for the benchmark problem.

Table 3. Different value of uncertainty in each Membership function with manually increase (15% 5% 10% 20% Increase respectively)

No.	N	n	Epsilon (ε)	Index	Data inside the interval output
Base	0.3418	20	0	0.1	67/192
1	0.6582	20	0.125	0.19	129/192
2	0.7602	20	0.25	0.22	149/192
3	0.7959	20	0.375	0.23	156/192
4	0.8112	20	0.5	0.23	159/192
5	0.8418	20	0.625	0.24	165/192
6	0.852	20	0.75	0.24	167/192
7	0.8571	20	0.875	0.24	168/192
8	0.8571	20	1	0.24	168/192
9	0.8571	20	1.125	0.24	168/192
10	0.8622	20	1.25	0.25	169/192
11	0.8622	20	1.375	0.25	169/192
12	0.8622	20	1.5	0.25	169/192
13	0.8622	20	1.625	0.25	169/192
14	0.8724	20	1.75	0.25	171/192
15	0.8724	20	1.875	0.25	171/192
16	**0.8878**	**20**	**2**	**0.25**	**174/192**
17	0.8673	20	2.125	0.25	170/192
18	0.8571	20	2.25	0.24	168/192
19	0.8469	20	2.375	0.24	166/192
20	0.8469	20	0.125	0.24	166/192

We can observe that in case two, where the increase of epsilon is different for each input, we get the best index with less increase of uncertainty in their membership functions for this problem.

4.2 Adaptive Noise Cancelation (ANC)

The simulation results based on the benchmark problem of Adaptive Noise Cancellation of a transmitted signal are presented in this section. The objective of ANC is to filter out an interference component by identifying a linear model between a measurable noise source and the corresponding unmeasurable interference. ANC using linear filters has been used successfully in real-world applications such as interference canceling in electrocardiograms, echo elimination on long-distance telephone transmission lines, and antenna sidelobe interference canceling. The original data base contains 601 data points [15].

For this benchmark problem we used a genetic algorithm, which we used a roulette wheel selection method, multi-point crossover and real-valued mutation.

Table 4 shows the simulations results with a GA for Equal Values of Uncertainty for all MF's for ANC benchmark problem.

Table 5, shows the simulations results with a GA for different values of uncertainty in each input for ANC benchmark problem.

Table 4. Equal Value of Uncertainty for all MF's

Individuals	Generation	Mutation	Crossover	Time Execution	Epsilon	Data inside the interval output
80	100	0.1	0.4	1:50:28	1.00183	524/601
150	200	0.3	0.7	6:36:42	1	525/601
100	100	0.001	0.5	2:13:45	1.02672	526/601
40	50	0.8	0.9	0:55:31	1	529/601
30	30	0.5	0.6	0:12:26	1.00441	530/601
50	50	1	1	0:33:36	1.00355	530/601
60	30	0.4	0.85	0:46:34	1	538/601
35	30	0.95	1	0:26:21	1	552/601
25	30	0.8	1	0:19:02	1	556/601
15	100	1	1	0:20:36	1	564/601
10	100	0.9	1	0:13:20	1	565/601
10	30	0.6	0.8	0:08:18	1.502	579/601
5	30	0.7	0.9	0:04:04	2	597/601
20	30	0.5	0.7	0:16:09	2	600/601
15	**30**	**0.35**	**0.85**	**0:12:11**	**3.4934**	**601/601**

Table 5. Different value of uncertainty in each input

Individuals	Generation	Mutation	Crossover	Time Execution	Epsilon First Input	Epsilon Second Input	Data inside the interval output
80	100	0.1	0.4	0:50:58	1.500	1.500	584/601
150	200	0.3	0.7	3:09:31	2.000	1.663	597/601
100	100	0.001	0.5	1:08:40	1.505	1.501	580/601
40	50	0.8	0.9	0:25:55	1.300	2.693	572/601
30	30	0.5	0.6	0:11:48	2.000	1.000	598/601
50	50	1	1	0:33:17	2.064	1.061	599/601
60	30	0.4	0.85	0:15:25	2.000	3.459	597/601
35	30	0.95	1	0:13:28	2.000	2.000	597/601
25	30	0.8	1	0:09:44	1.500	2.500	597/601
15	100	1	1	0:10:09	2.000	1.167	599/601
10	100	0.9	1	0:06:37	1.500	1.520	588/601
10	30	0.6	0.8	0:04:07	1.000	1.000	593/601
5	30	0.7	0.9	0:01:57	1.001	1.000	582/601
20	**30**	**0.5**	**0.7**	**0:07:56**	**2.000**	**1.534**	**601/601**
15	30	0.35	0.85	0:05:45	1.000	1.500	567/601

Table 6. Different value of uncertainty in each Membership function

Individuals	Generation	Mutation	Cross	Time Execution	Epsilon 1th MF 1th Input	Epsilon 2th MF 1th Input	Epsilon 1th MF 2th Input	Epsilon 2th MF 2th Input	Data inside the interval output
80	100	0.1	0.4	2:01:39	2.000	1.001	1.000	1.000	592/601
150	150	0.3	0.7	5:34:46	1.500	2.000	3.981	3.713	597/601
100	100	0.001	0.5	2:16:09	1.510	2.048	3.906	3.571	593/601
40	50	0.8	0.9	0:24:59	2.000	1.588	3.237	2.332	598/601
30	30	0.5	0.6	0:11:23	1.500	2.000	3.747	3.500	601/601
50	50	1	1	0:33:06	2.000	1.000	1.009	1.000	595/601
60	30	0.4	0.85	0:23:07	2.000	1.000	1.012	1.047	592/601
35	30	0.95	1	0:14:47	4.000	3.860	3.519	4.000	601/601
25	30	0.8	1	0:09:40	2.003	3.625	4.000	2.600	601/601
15	100	1	1	0:20:56	1.000	1.000	1.025	1.000	560/601
10	100	0.9	1	0:13:15	2.031	2.000	3.465	1.830	576/601
10	**30**	**0.6**	**0.8**	**0:04:00**	**1.006**	**2.000**	**3.252**	**2.725**	**601/601**
5	30	0.7	0.9	0:01:56	1.047	1.698	1.640	1.523	600/601
20	30	0.5	0.7	23:07:16	3.963	4.000	1.979	1.981	601/601
15	30	0.35	0.85	0:05:58	1.000	2.943	3.997	3.594	601/601

Table 6, shows the simulations results with a GA for different values of uncertainty in each Membership function for ANC benchmark problem.

We can observe that the best result in ANC benchmark problem with a GA is for different value of uncertainty in each input, where we have the total data inside the interval output with a less average value of epsilon.

5 Conclusions

We presented in this paper a description of an optimization method for the membership functions of type-2 fuzzy systems based on the level of uncertainty. Simulation results for the Miles per Gallon and Adaptive Noise Cancellation benchmark problems are presented. For ANC problem we can see that varying the uncertainty of the membership functions with the GA changes the fitness of the fuzzy systems and the optimal fuzzy system can be obtained.

Acknowledgements

We would like to express our gratitude to CONACYT under grant number 175883, UABC University and Tijuana Institute of Technology for the facilities and resources for the development of this research.

References

1. Melin, P., Castillo, O., Gómez, E., Kacprzyk, J., Pedrycz, W.: Analysis and Design of Intelligent Systems Using Soft Computing Techniques. Advances in Soft Computing, vol. 41. Springer, Heidelberg (2007)
2. Melin, P., Castillo, O., Kacprzyk, J., Pedrycz, W.: Hybrid Intelligent Systems. Studies in Fuzziness and Soft Computing (Hardcover - December 20, 2006)
3. Melin, P., Castillo, O., Gómez, E., Kacprzyk, J.: Analysis and Design of Intelligent Systems using Soft Computing Techniques. Advances in Soft Computing (Hardcover - July 11, 2007)

4. Mendel, J.M.: Uncertain Rule-Based Fuzzy Logic Systems for Wireless Communications. In: FUZZ-IEEE 2001 (2001)
5. Karnik, N.N., Mendel, J.M.: Operations on type-2 fuzzy sets. Fuzzy Sets and Systems 122(2), 327–348 (2001)
6. Zadeh, L.A.: Fuzzy Logic. Computer 1(4), 83–93 (1998)
7. Zadeh, L.A.: Knowledge representation in Fuzzy Logic. IEEE Transactions on Knowledge Data Engineering 1, 89 (1989)
8. Zadeh, L.A.: Fuzzy Logic = Computing with Words. IEEE Transactions on Fuzzy Systems 4(2), 103 (1996)
9. Hidalgo, D., Melin, P., Castillo, O.: Type-1 and Type-2 Fuzzy Inference Systems as Integration Methods in Modular Neural Networks for Multimodal Biometry and its Optimization with Genetic Algorithms. Journal of Automation, Mobile Robotics & Intelligent Systems 2(1) (2008) ISSN 1897-8649
10. Castro, J.R., Castillo, O., Melin, P.: An Interval Type-2 Fuzzy Logic Toolbox for Control Applications. In: FUZZ-IEEE 2007, pp. 1–6 (2007)
11. Mendel, J.M.: UNCERTAIN Rule-Based Fuzzy Logic Systems, Introduction and New Directions. Prentice Hall, Englewood Cliffs (2001)
12. Mendel, J.M., Bob-John, R.I.: Type-2 Fuzzy Sets Made Simple. IEEE Transactions on Fuzzy Systems 10(2), 117 (2002)
13. Karnik, N., Mendel, J.M.: Operations on type-2 fuzzy sets, Signal and Image Processing Institute, Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA, USA (May 11, 2000)
14. Man, K.F., Tang, K.S., Kwong, S.: Genetic Algorithms. Concepts and Designs. Springer, Heidelberg (1999)
15. Jang, J.-S.R., Sun, C.-T., Mizutani, E.: Neuro-Fuzzy and Soft Computing, A Computational Approach to Learning and Machine Intelligence. Prentice Hall, Englewood Cliffs (1997)
16. Castillo, O., Melin, P.: Type-2 Fuzzy Logic: Theory and Applications. Studies in Fuzziness and Soft Computing. Springer, Heidelberg (2008) ISSN 1434-9922

Fuzzy Logic for Parameter Tuning in Evolutionary Computation and Bio-inspired Methods

Fevrier Valdez, Patricia Melin, and Oscar Castillo

Computer Science in the Graduate Division, Tijuana Institute of Technology, Tijuana, B.C.
fevrier@tectijuana.mx, pmelin@tectijuana.mx,
ocastillo@tectijuana.mx

Abstract. We describe in this paper an approach for mathematical function optimization using fuzzy logic for parameter tuning combining Particle Swarm Optimization (PSO) and Genetic Algorithms (GAs). The proposed method combines the advantages of PSO and GA to give us an improved FPSO+FGA hybrid method. Fuzzy logic is helpful to find the optimal parameters in PSO and GA in the best way possible. Also, with the tuning of parameters based on fuzzy logic it is possible to balance the exploration and exploitation of the proposed method. The hybrid method is called FPSO+FGA and was tested with a set of benchmark mathematical functions.

Keywords: FPSO, FGA, Fuzzy Logic.

1 Introduction

We describe in this paper an evolutionary method combining PSO and GA, to give us an improved FPSO+FGA hybrid method. We apply the hybrid method to mathematical function optimization to validate the new approach. In this case, we are using a set of mathematical benchmark functions [4][5][13] to compare the optimization results among a GA, PSO and FPSO+FGA.

Several approaches had been proposed for PSO and GA, for example, in [15] can be seen an approach with GA and PSO for control vector for loss minimization of induction motor. In [16] it can be seen an approach with PSO, GA and Simulated Annealing (SA), for scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm.

The main motivation of this method is to combine the characteristics of a GA and PSO [1][2]. We are using several fuzzy systems to perform dynamical parameter adaptation. For decision making between the methods depending on the results that we are generating we are using another fuzzy system. The paper is organized as follows: in section 2 a description of Genetic Algorithms for optimization problems is presented, in section 3 the Particle Swarm Optimization is presented, in section 4 the proposed method FPSO+FGA, mathematical description and the fuzzy systems are described, in section 5 the experimental results are described, and finally in section 6 the conclusions obtained after the study of the proposed evolutionary computing methods are presented.

2 Genetic Algorithms for Optimization

Holland, from the University of Michigan initiated his work on genetic algorithms at the beginning of the 1960s. His first achievement was the publication of *Adaptation in Natural and Artificial System* [7] in 1975.

He had two goals in mind: to improve the understanding of natural adaptation process, and to design artificial systems having properties similar to natural systems [8].

The basic idea is as follows: the genetic pool of a given population potentially contains the solution, or a better solution, to a given adaptive problem. This solution is not "active" because the genetic combination on which it relies is split between several subjects. Only the association of different genomes can lead to the solution.

Holland's method is especially effective because it not only considers the role of mutation, but it also uses genetic recombination, (crossover) [9]. The essence of the GA in both theoretical and practical domains has been well demonstrated [1]. The concept of applying a GA to solve engineering problems is feasible and sound. However, despite the distinct advantages of a GA for solving complicated, constrained and multiobjective functions where other techniques may have failed, the full power of the GA in application is yet to be exploited [12] [14].

The Simple Genetic Algorithm can be expressed in pseudo code with the following cycle:

```
1. Generate the initial population of individuals aleatorily P(0).
2. While (number _ generations <= maximum _ numbers _ generations)
     Do:
       {
         Evaluation;
         Selection;
         Reproduction;
         Generation ++;
       }
3. Show results
   4. End
```

3 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995, inspired by the social behavior of bird flocking or fish schooling [3].

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA) [6]. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike the GA, the PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles [10].

Each particle keeps track of its coordinates in the problem space, which are associated with the best solution (fitness) it has achieved so far (The fitness value is also stored). This value is called *pbest*. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called *lbest*. When a particle takes all the population as its topological neighbors, the best value is a global best and is called *gbest*.

The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its p*best* and *lbest* locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *lbest* locations.

In the past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods [11].

The pseudo code of the PSO is as follows:

```
 For each particle
 Initialize particle
End
Do
 For each particle
  Calculate fitness value
  If the fitness value is better than the best fitness value (pBest) in history
   set current value as the new pBest
 End
 Choose the particle with the best fitness value of all the particles as the gBest
 For each particle
  Calculate particle velocity
  Update particle position
 End
While maximum iterations or minimum error criteria is not attained
```

4 FPSO+FGA Method

The general approach of the proposed method PSO+GA can be seen in Figure 2. The method can be described as follows:

1. It receives a mathematical function to be optimized
2. It evaluates the role of both GA and PSO.
3. A main fuzzy system is responsible for receiving values resulting from step 2.
4. The main fuzzy system decides which method to use(GA or PSO)
5. Another fuzzy system receives the Error and DError as inputs to evaluates if is necessary change the parameters in GA or PSO.
6. There are 3 fuzzy systems. One is for decision making (is called main fuzzy), the second one is for changing parameters of the GA (is called fuzzyga) in this case change the value of crossover (k_1) and mutation (k_2) and the third fuzzy system is used to change parameters of the PSO (is called fuzzypso) in this case change the value of social acceleration (c_1) and cognitive acceleration (c_2).
7. The main fuzzy system decides in the final step the optimum value for the function introduced in step 1. Repeat the above steps until the termination criterion of the algorithm is met.

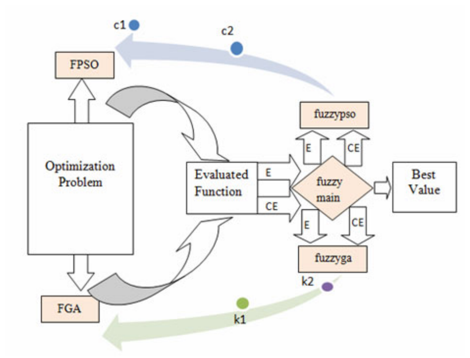
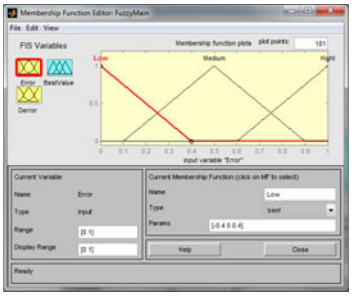


Fig. 1. The FPSO+FGA scheme

The basic idea of the FPSO+FGA scheme is to combine the advantages of the individual methods using a fuzzy system for decision making and the others two fuzzy systems to improve the parameters of the FGA and FPSO when is necessary.

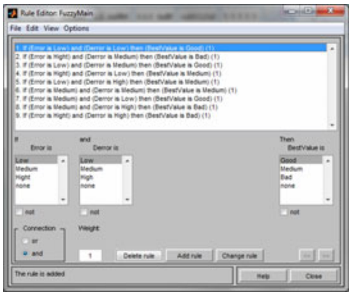
As can be seen in the proposed hybrid FPSO+FGA method, it is the internal fuzzy system structure, which has the primary function of receiving as inputs (Error and DError) the results of the FGA and FPSO outputs. The fuzzy system is responsible for integrating and decides which are the best results being generated at run time of the FPSO+FGA. It is also responsible for selecting and sending the problem to the "fuzzypso" fuzzy system when the FPSO is activated or to the "fuzzyga" fuzzy system when FGA is activated. Also activating or temporarily stopping depending on the results being generated. Figure 3 shows the membership functions of the main fuzzy system that is implemented in this method. The fuzzy system is of Mamdani type because it is more common in this type of fuzzy control and the defuzzification method is the centroid. In this case, we are using this type of defuzzification because in other papers we have achieved good results [4]. The membership functions are of triangular form in the inputs and outputs as is shown in Figure 2. Also, the membership functions were chosen of triangular form based on past experiences in this type of fuzzy control. The fuzzy system consists of 9 rules. For example, one rule is if error is Low and DError is Low then best value is Good (view Figure 3). Figure 4 shows the fuzzy system rule viewer. Figure 5 shows the surface corresponding to this fuzzy system. The other two fuzzy systems are similar to the main fuzzy system.



Current Variable
Name Error
Type input
Range [0 1]
Display Range [0 1]

Current Membership Function (click on MF to select)
Name Low
Type trimf
Params [-0.4 0 0.4]

Fig. 2. Membership functions of the fuzzy system



1. If (Error is Low) and (Derror is Low) then (BestValue is Good) (1)
2. If (Error is Hight) and (Derror is Medium) then (BestValue is Bad) (1)
3. If (Error is Low) and (Derror is Medium) then (BestValue is Good) (1)
4. If (Error is Hight) and (Derror is Low) then (BestValue is Medium) (1)
5. If (Error is Low) and (Derror is High) then (BestValue is Medium) (1)
6. If (Error is Medium) and (Derror is Medium) then (BestValue is Medium) (1)
7. If (Error is Medium) and (Derror is Low) then (BestValue is Good) (1)
8. If (Error is Medium) and (Derror is High) then (BestValue is Bad) (1)
9. If (Error is Hight) and (Derror is High) then (BestValue is Bad) (1)

The rule is added

Fig. 3. Rules of the fuzzy system

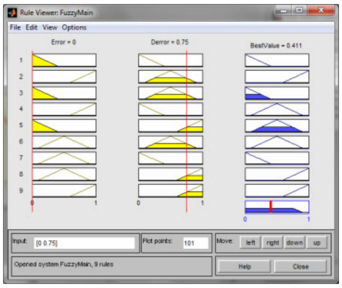


Fig. 4. Rule viewer for the fuzzy system

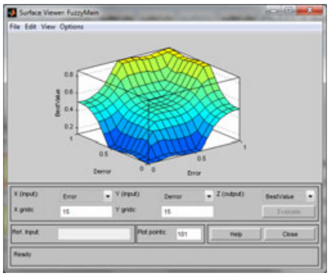


Fig. 5. Surface of the main fuzzy system

4.1 Mathematical Description of the FPSO+FGA

This section describes the formal mathematical definition of the proposed method FPSO + FGA. The mathematical description for the FPSO is defined as follows:

Equation 1 shows a fundamental part of traditional PSO algorithm; were c_1 and c_2 are represented by constants values. In equation 2 it is described the way in which the basic equation of PSO is modified to achieve the goal, and then, convert part of it in fuzzy parameters, we can see the differences between two equations that c_1 and c_2 in equation 2 are those that change, because an essential part of the proposed method lies in those two variables. Traditionally these two variables are constant, in this case, the importance of these two accelerations, is that we decided to obtain these two as fuzzy parameters.

$$v_{ij}(t+1) = vi_j(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \tag{1}$$

Where $vi_j(t)$ is the velocity of particle i in dimension j = 1,…,n_x at time step t, $xi_j(t)$ is the position of particle i in dimension j at time step t, c_1 and c_2 represents the cognitive and social acceleration. In this case, these values are fuzzy because they are changing dynamically when the FPSO is running, and are defined by expressions 3 and 4, and r_{1j} (t), r_{2j} ~U(0,1) are random values in the range [0,1].

$$v_{ij}(t+1) = vi_j(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \tag{2}$$

$$c_1 = \frac{\sum_{i=1}^{r_{c_1}} \mu_i^{c_1}(c_{1i})}{\sum_{i=1}^{r_{c_1}} \mu_i^{c_1}} \tag{3}$$

Where: c_1 = Percentage of cognitive acceleration of the particle *i*.

r_{c1} = Number of rules of the fuzzy system corresponding to c_1.

c_{1i} = Output result for rule *i* corresponding to c_1.

$\mu_i^{c_1}$ = Membership function of rule *i* corresponding to c_1.

$$c_2 = \frac{\sum_{i=1}^{r_{c_2}} \mu_i^{c_2}(c_{2i})}{\sum_{i=1}^{r_{c_2}} \mu_i^{c_2}} \tag{4}$$

Where: c_2 = Percentage of cognitive acceleration of the particle *i*.

r_{c_2} = Number of rules of the fuzzy system corresponding to c_1.

c_{2i} = Output result for rule *i* corresponding to c_1.

$\mu_i^{c_2}$ = Membership function of rule *i* corresponding to c_1.

For the FGA the mathematical description is defined as follows:
Several crossover operators have been developed for GAs, depending on the format in which individuals are represented. For binary representations, uniform crossover, one point crossover and two points cross over are the most popular. In this case we are using two points crossover with a fuzzy crossover rate because we are adding a fuzzy system called 'fuzzyga' that is able of change the crossover and mutation rate.

As in the FPSO, also in the FGA, we are tuning the parameters with a fuzzy system; in this case, the parameters adjusted are the crossover (k_1) and mutation (k_2). The expressions 5 and 6 show how the values for k_1 and k_2 were obtained.

$$k_1 = \frac{\sum_{i=1}^{r_{k_1}} \mu_i^{k_1}(k_{1i})}{\sum_{i=1}^{r_{k_1}} \mu_i^{k_1}} \tag{5}$$

Where: k_1 = Percentage of crossover of the particle *i*.

r_{k_1} = Number of rules of the fuzzy system corresponding to k_1.

k_{1i} = Output result for rule *i* corresponding to k_1.

$\mu_i^{k_1}$ = Membership function of rule *i* corresponding to k_1.

$$k_1 = \frac{\sum_{i=1}^{r_{k_1}} \mu_i^{k_1}(k_{1i})}{\sum_{i=1}^{r_{k_1}} \mu_i^{k_1}} \tag{6}$$

Where: k_2 = Percentage of mutation of the particle *i*.

r_{k_2} = Number of rules of the fuzzy system corresponding to k_2.

k_{2i} = Output result for rule *i* corresponding to k_2.

$\mu_i^{k_1}$ = Membership function of the rule *i* corresponding to k_2.

4.2 Definition of the Fuzzy Systems Used in FPSO+FGA

'fuzzypso': In this case we are using a fuzzy system called 'fuzzypso', and the structure of this fuzzy system is as follows:

Number of Inputs: 2
Number of Outputs: 2
Number of membership functions: 3
Type of the membership functions: Triangular
Number of rules: 9
Defuzzification: Centroid
The main function of the fuzzy system called 'fuzzypso' is to adjust the parameters of the PSO. In this case, we are adjusting the following parameters: 'c_1' and 'c_2' ; where:
'c_1' = Cognitive Acceleration
'c_2' = Social Acceleration
We are changing these parameters to test the proposed method. In this case, with 'fuzzypso' is possible to adjust in real time the 2 parameters that belong to the PSO.
'fuzzyga': In this case we are using a fuzzy system called **'fuzzyga'**, the structure of this fuzzy system is as follows:
Number of Inputs: 2
Number of Outputs: 2
Number of membership functions: 3

Type of membership functions: Triangular
Number of rules: 9
Defuzzification: Centroid
The main function of the fuzzy system called ‘fuzzypso’ is to adjust the parameters of the GA. In this case, we are adjusting the following parameters: ‘k_1’, ‘k_2’; where:
‘k_1’ = mutation
‘k_2’ = crossover
‘fuzzymain’: In this case, we are using a fuzzy system called **‘fuzzymain’**. The structure of this fuzzy system is as follows:
Number of Inputs: 2
Number of Outputs: 1
Number of membership functions: 3
Type of membership functions: Triangular
Number of rules: 9
Defuzzification: Centroid
The main function of the fuzzy system, called ‘fuzzymain’ is to decide on the best way for solving the problem, in other words if it is more reliable to use the FPSO or FGA. This fuzzy system is able to receive two inputs, called error and derror, it is to evaluate the results that are generated by FPSO and FGA in the last step of the algorithm.

5 Experimental Results

To validate the proposed method we used a set of 5 benchmark mathematical functions; all functions were evaluated with different numbers of dimensions, in this case, the experimental results were obtained with 32, 64 and 128 dimensions.

Table 1 shows the definitions of the mathematical functions used in this paper. The global minimum for the test functions is 0.

Table 1. Mathematical functions

Function	Definition	Domain
De Jong’s	$f_1=\sum_{n=i}^{N} x_n^2$	- ∞≤x≤∞
Rotated Hyper-Ellipsod	$f(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	- 65≤xi≤ 65
Rosenbrock’s Valley	$f(x)=\sum_{i=1}^{n-1} 100.\left(x_{i+1}-x_i^2\right)^2+(1-x_i)^2$	- 2.≤xi≤2
Rastrigin’s	$f(x)=10.n+\sum_{i=1}^{n}\left(x_i^2-10.\cos\left(2\pi x_i\right)\right)$	- ∞≤xi≤ ∞
Griewank’s	$f(x)=\sum_{i=1}^{n}\frac{x_i^2}{4000}-\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	- ∞≤xi≤ ∞

Tables 2, 3 and 4 show the experimental results for the benchmark mathematical functions used in this research. The table shows the experimental results of the evaluations for each function with 32, 64 and 128 dimensions; where it can be seen the best and worst values obtained, and the average of 50 times after executing the method.

Table 2. Experimental results with 32 dimensions

Function	Average	Best	Worst
De Jong's	0.0111	0.0299	1.5926
Rotated Hyper- Ellipsod	0.9970	0.0690	8.4104
Rosenbrock's Valley	1.0023	0.3422	9.8790
Rastrigin's	0.7890	0.0012	3.3345
Griewank's	0.8801	0.1045	4.5678

Table 3. Experimental results with 64 dimensions

Function	Average	Best	Worst
De Jong's	0.2529	0.0123	1.79
Rotated Hyper- Ellipsod	4.0255	2.1667	42.872
Rosenbrock's Valley	3.0568	2.5340	7.7765
Rastrigin's	2.3443	0.9766	5.6666
Griewank's	2.0967	0.9981	6.4561

Table 4. Simulations results with 128 dimensions

Function	Average	Best	Worst
De Jong's	0.2060	0.1681	2.089
Rotated Hyper- Ellipsod	4.9908	3.0999	78.09
Rosenbrock's Valley	6.0676	2.9909	9.0456
Rastrigin's	3.4433	1.0100	10.098
Griewank's	4.3245	1.5567	12.980

6 Conclusions

The analysis of the experimental results of the evolutionary method considered in this paper, the FPSO+FGA, lead us to the conclusion that for the optimization of this benchmark mathematical function this method is a good alternative, because it is easier and very fast to optimize and achieve good results than to try it with PSO or GA separately [5], especially when the number of dimensions is increased. This is, because the combination PSO and GA with fuzzy rules gives a hybrid method FPSO+FGA.

Acknowledgment

We would like to express our gratitude to the CONACYT, Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

References

1. Man, K.F., Tang, K.S., Kwong, S.: Genetic Algorithms: Concepts and Designs. Springer, Heidelberg (1999)
2. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan, pp. 39–43 (1995)
3. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942–1948 (1995)
4. Holland, J.H.: Adaptation in natural and artificial system. The University of Michigan Press, Ann Arbor (1975)
5. Valdez, F., Melin, P.: Parallel Evolutionary Computing using a cluster for Mathematical Function Optimization, Nafips, San Diego, CA, USA, pp. 598–602 (June 2007)
6. Castillo, O., Melin, P.: Hybrid intelligent systems for time series prediction using neural networks, fuzzy logic, and fractal theory. IEEE Transactions on Neural Networks 13(6), 1395–1408 (2002)
7. Fogel, D.B.: An introduction to simulated evolutionary optimization. IEEE transactions on neural networks 5(1), 3–14 (1994)
8. Goldberg, D.: Genetic Algorithms. Addison-Wesley, Reading (1988)
9. Emmeche, C.: Garden in the Machine. The Emerging Science of Artificial Life, p. 114. Princeton University Press, Princeton (1994)
10. Angeline, P.J.: Using Selection to Improve Particle Swarm Optimization. In: Proceedings 1998 IEEE World Congress on Computational Intelligence, Anchorage, Alaska, pp. 84–89. IEEE, Los Alamitos (1998)
11. Angeline, P.J.: Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 601–610. Springer, Heidelberg (1998)
12. Back, T., Fogel, D.B., Michalewicz, Z. (eds.): Handbook of Evolutionary Computation. Oxford University Press, Oxford (1997)
13. Montiel, O., Castillo, O., Melin, P., Rodriguez, A., Sepulveda, R.: Human evolutionary model: A new approach to optimization. Inf. Sci. 177(10), 2075–2098 (2007)
14. Castillo, O., Valdez, F., Melin, P.: Hierarchical Genetic Algorithms for topology optimization in fuzzy control systems. International Journal of General Systems 36(5), 575–591 (2007)
15. Kim, D., Hirota, K.: Vector control for loss minimization of induction motor using GA–PSO. Applied Soft Computing 8, 1692–1702 (2008)
16. Liu, H., Abraham, A.: Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. Future Generation Computer Systems (article in press)

Fuzzy Logic Controllers Optimization Using Genetic Algorithms and Particle Swarm Optimization

Ricardo Martinez-Soto[1], Oscar Castillo[2], Luis T. Aguilar[3], and Patricia Melin[2]

[1] School of Engineering UABC University,
Tijuana, México
mc.ricardo.martinez@hotmail.com
[2] Division of Graduate Studies Tijuana Institute of Technology,
Tijuana, México
ocastillo@tectijuana.mx, pmelin@tectijuana.mx
[3] Centro de Investigación y Desarrollo de Tecnología Digital,
Instituto Politécnico Nacional, Tijuana, México
luis.aguilar@ieee.org

Abstract. In this paper we apply to Bio-inspired and evolutionary optimization methods to design fuzzy logic controllers (FLC) to minimize the steady state error of linear systems. We test the optimal FLC obtained by the genetic algorithms and the PSO applied on linear systems using benchmark plants. The bio-inspired and the evolutionary methods are used to find the parameters of the membership functions of the FLC to obtain the optimal controller. Simulation results are obtained with Simulink showing the feasibility of the proposed approach.

Keywords: Fuzzy Logic Controllers, Genetic Algorithms, Particle Swarm Optimization.

1 Introduction

Optimization is a term used to refer to a branch of computational science concerned with finding the "best" solution to a problem. Here, "best" refers to an acceptable (or satisfactory) solution, which may be the absolute best over a set of candidate solutions, or any of the candidate solutions. The characteristics and requirements of the problem determine whether the overall best solution can be found [1]. Optimization algorithms are search methods, where the goal is to find a solution to an optimization problem, such that a given quantity is optimized, possibly subject to a set of constraints [1],[2],[3]. Some optimization methods are based on populations of solutions. Unlike the classic methods of improvement for trajectory tracking, in this case each iteration of the algorithm has a set of solutions. These methods are based on generating, selecting, combining and replacing a set of solutions. Since they maintain and they manipulate a set, instead of a unique solution throughout the entire search process, they use more computer time than other metaheuristic methods. This fact can be aggravated because the "convergence" of the population requires a great number of iterations. For this reason a concerted effort has been dedicated to obtaining methods that are more aggressive and manage to obtain solutions of quality in a nearer horizon. This paper is concerned with bio-inspired optimization methods like genetic algorithms (GA) and particle swarm optimization (PSO) to design optimized fuzzy logic controllers (FLC) for linear and non-linear problems. The bio-inspired methods are used to find the parameters of the membership functions obtaining the optimal FLC for plant control.

This paper is organized as follows: Section 2 presents the theoretical basis and problem statement. Section 3 introduces the controller design where a GA and PSO are used to select the parameters. Robustness properties of the closed-loop system are achieved with a fuzzy logic control system using a Takagi-Sugeno model where the error and the change of error, are considered as the linguistic variables. Section 4 provides a simulation study of linear systems using the controller described in Section 3. Finally, Section 5 presents the conclusions.

2 Theoretical Basis and Problem Statement

2.1 Particle Swarm Optimization (PSO)

PSO is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling [4]. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA) [5]. The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike the GA, the PSO has no evolution operators such as crossover and mutation. In the PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles [6]. Each particle keeps track of its coordinates in the problem space, which are associated with the best solution (fitness) it has achieved so far (The fitness value is also stored). This value is called *pbest*. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called *lbest*. When a particle takes all the population as its topological neighbors, the best value is a global best and is called *gbest* [7].

The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its p*best* and *lbest* locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *lbest* locations [6]. In the past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods [8], [9]. Another reason that PSO is attractive is that there are few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement [8].

2.2 Genetic Algorithms (GA)

Genetic Algorithms (GAs) are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection and genetic processes [10]. The basic principles of GAs were first proposed by John Holland in 1975, inspired by the mechanism of natural selection where stronger individuals are likely the winners in a competing environment [11], [12], [13]. GA presumes that the potential solution of any problem is an individual and can be represented by a set of parameters. These parameters are regarded as the genes of a chromosome and can be structured by a string of values in binary form. A positive value, generally know as a fitness value, is used to reflect the degree of "goodness" of the chromosome for the problem which would be highly related with its objective value. The GA works as follows:

1. Start with a randomly generated population of n chromosomes (candidate solutions to a problem).
2. Calculate the fitness of each chromosome in the population.
3. Repeat the following steps until *n* offspring have been created:
    a. Select a pair of parent chromosomes from the current population, the probability of selection being an increasing function of fitness. Selection is done with replacement, meaning that the same chromosome can be selected more than once to become a parent.
    b. With probability (crossover rate), perform crossover to the pair at a randomly chosen point to a form two offspring.
    c. Mutate the two offspring at each locus with probability (mutation rate), and place the resulting chromosomes in the new population.
4. Replace the current population with the new population.
5. Go to step 2.

The simple procedure just described above is the basis for most applications of GAs.

2.3 Problem Statement

To test the optimized FLC's obtained with the bio-inspired methods; we used different linear systems. We consider two benchmark problems called Plant 1 and Plant 2 with different levels of complexity.

Plant 1 is given by the following second order transfer function:

$$g(s) = \frac{w_n^2}{s^2 + 2\varepsilon\, w_n s + w_n^2} \qquad \varepsilon = 0.5, \quad W_n = 2 \tag{1}$$

where w_n is the natural frequency and ε is the coefficient damping.

Plant 2 is given by the following transfer function:

$$g(s) = \frac{1}{s^2 + 4} \tag{2}$$

3 Fuzzy Logic Control Design

In this section we design a fuzzy logic controller (FLC) where the optimal controller was found with the GAs and the PSO. For the FLC a Takagi-Sugeno type of fuzzy system is used with two inputs a) error, and b) error change, with three membership functions for each input, “Negative, Zero and Positive” (Gaussian and triangular), and one output, defined with constant values [14],[15],[16],[17],[18],[19],[20],[21]. Fig 1 shows the FLC membership functions for the plant control and Table I show the Fuzzy Rules.

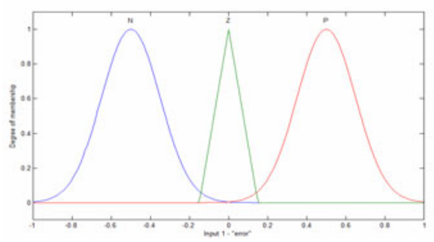


Table 1. Fuzzy rules of the FLC

	Negative	Zero	Positive
Negative	N	N	Z
Zero	N	Z	P
Positive	Z	P	P

Fig. 1. Gaussian and Triangular membership functions and table 1 show the fuzzy rules

Once we obtained the FLC design, we set the parameters of both methods: the GA chromosome has 17 genes of real values that represent the two inputs, error and error change and one output constant values and using different values in the genetic operator’s; mutation and single point crossover. We use different values for the cognitive, social parameters of the PSO and inertial value to balance the swarm. Both bio-inspired methods use the same space of solutions (population) that we use to find the optimal values of the parameters of the membership functions. Table 2 shows the parameters of the membership functions, the minimal and the maximum values in the search range for the GA and PSO to find the best fuzzy controller system for the linear plants.

Table 2. Parameters of the Membership Functions

Plant 1				Plant 2			
MF Type	Point	Min Value	Max Value	MF Type	Point	Min Value	Max Value
Gauss	a	0.3	0.6	Gauss	a	1.8	2.8
	b	-1	-1		b	-05	-5
	a	-0.3	-0.8		a	-0.5	-4
Triang	b	0	0	Triang	b	0	0
	c	0.3	0.8		c	0.5	4
Gauss	a	0.3	0.6	Gauss	a	1.8	2.8
	b	1	1		b	5	5

4 Simulation Results

In this section, we evaluate, through computer simulations performed in MATLAB® and SIMULINK®, the designed FLC for the benchmark problems (Plant 1 and Plant 2). We changed the parameters values of the GA to find our best FLC, using the maximum number of generations to stop the method. In PSO we only change the values of the cognitive a social constants to find the best FLC, and we used the maximum number of iterations to stop the method.

4.1 Plant 1 Using the GA

Table 3 presents the main results of the FLC obtained with genetic algorithms showing in the ninth row the best result.

Table 3. Results of the Type-1 FLC Obtained by genetic algorithms

No	Indiv	Gen	% Remp	Cross	Mut	GA Time	Average error
1	90	75	0.7	0.5	0.2	0:19:48	0.03667
2	120	70	0.7	0.5	0.1	0:23:36	0.04236
3	90	75	0.7	0.5	0.2	0:20:25	0.07037
4	90	35	0.7	0.5	0.2	0:09:10	0.07081
5	50	45	0.7	0.7	0.1	0:06:45	0.07682
6	40	25	0.7	0.7	0.4	0:03:00	0.08112
7	45	30	0.7	0.6	0.2	0:04:03	0.08260
8	70	45	0.7	0.5	0.1	0:10:08	0.08757
9	**150**	**80**	**0.7**	**0.7**	**0.1**	**0:33:25**	**0.08868**
10	20	15	0.7	0.8	0.3	0:00:54	0.09151

Fig. 2 shows the membership functions of input 1 and input 2 obtained by the genetic algorithm and Fig 3 shows the evolution of the genetic algorithm giving the best FLC for controlling the plant.

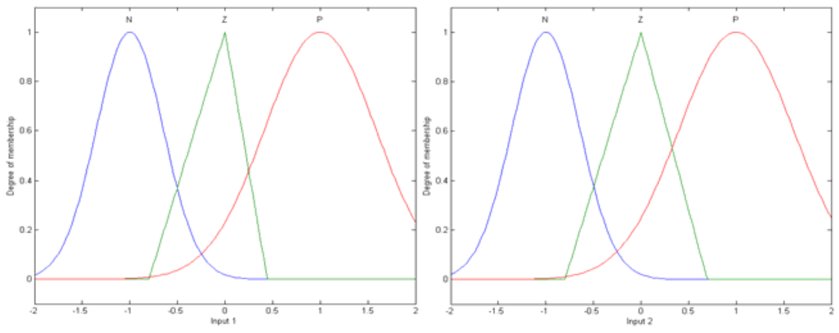


Fig. 2. Input 1 and input 2 membership functions of the optimized FLC

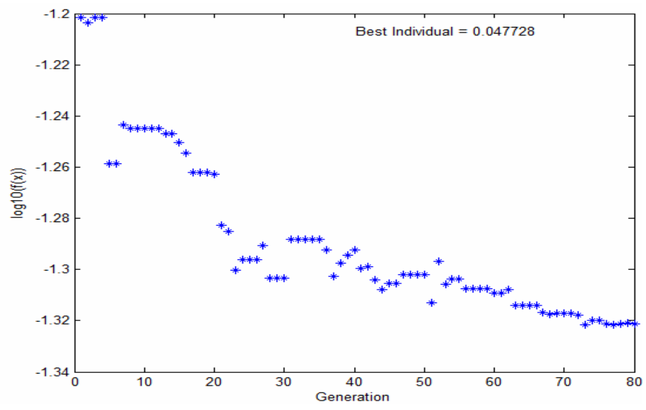


Fig. 3. Evolution of the GA for the FLC optimization

Fig. 4. Shows the control result of plant 1 using the optimized FLC obtained with the GA.

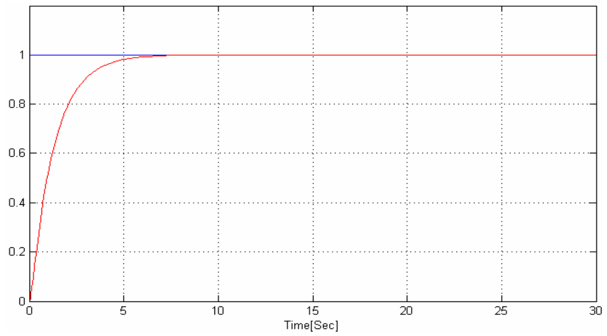


Fig. 4. Closed-loop response of plant 1 with the optimized FLC

4.2 Plant 1 Using the PSO

Table 4 presents the main results of the FLC obtained with the PSO method showing in the fourth row the best result.

Table 4. Results of the FLC Obtained by PSO

No.	Swarm	Max Iter	C1	C2	Inertia	Time exec	Average error
1	200	70	1	1	1	0:05:27	0.06321
2	200	70	0.5	0.5	1	0:07:35	0.05276
3	200	70	0.25	0.25	1	0:15:35	0.05576
4	**200**	**70**	**0.15**	**0.15**	**1**	**0:19:15**	**0.09717**
5	200	70	0.05	0.05	1	0:26:10	0.09400
6	200	70	0.005	0.005	1	0:26:56	0.11377

Fig 5 shows the membership functions of input 1 and input 2 obtained by the PSO and Fig 6 shows the particles behavior of the PSO giving the best FLC for controlling the plant 1.

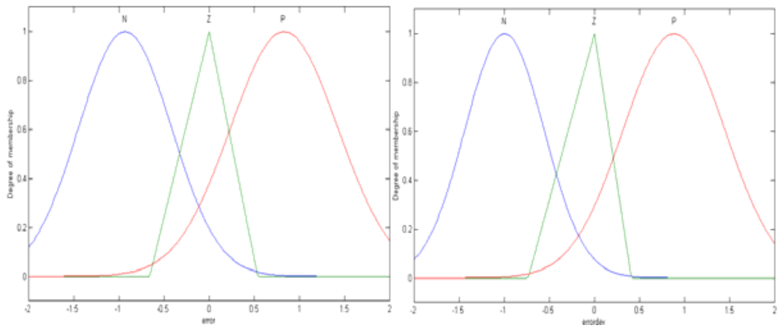


Fig. 5. Input 1 and input 2 membership functions of the optimized FLC

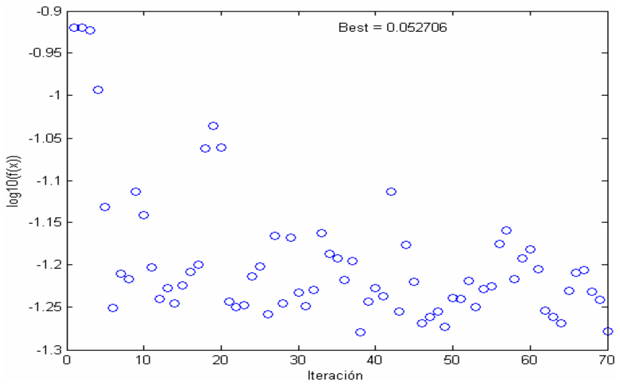


Fig. 6. Particles behavior of PSO

Fig. 7 Shows the control result of plant 1 using the optimized FLC obtained with the PSO.

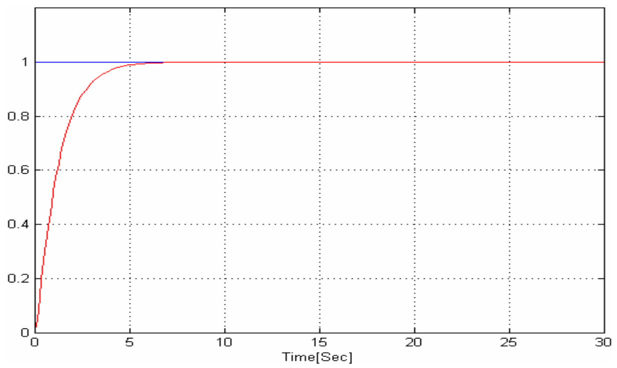


Fig. 7. Closed-loop response of plant 1 with the optimized FLC

4.3 Plant 2 Using the GA

Table 5 presents the main results of the FLC obtained with the genetic algorithms showing in the seven row the best result.

Table 5. Results of the FLC Obtained by Genetic algorithms for Plant 2

No	Indiv	Gen	% Remp	Cross	Mut	GA Time	Average error
1	120	70	0.7	0.5	0.1	0:22:30	0.10676
2	70	45	0.7	0.5	0.1	0:10:42	0.10681
3	150	80	0.7	0.7	0.1	0:31:09	0.10723
4	90	35	0.7	0.5	0.2	0:10:43	0.10918
5	90	75	0.7	0.5	0.2	0:23:01	0.10935
6	90	75	0.7	0.5	0.2	0:17:57	0.10969
7	**45**	**30**	**0.7**	**0.6**	**0.2**	**0:04:32**	**0.11193**
8	40	25	0.7	0.7	0.4	0:03:17	0.11784
9	50	45	0.7	0.7	0.1	0:07:51	0.11831
10	20	15	0.7	0.8	0.3	0:01:02	0.12315

Fig 8 shows the membership functions of input 1 and input 2 obtained by the GA and Fig 9 shows the evolution of the GA giving the best FLC for controlling the plant 2.

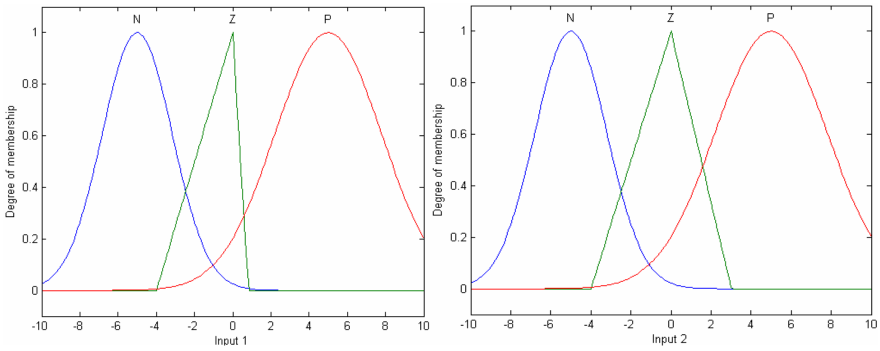


Fig. 8. Input 1 and input 2 membership functions of the optimized FLC

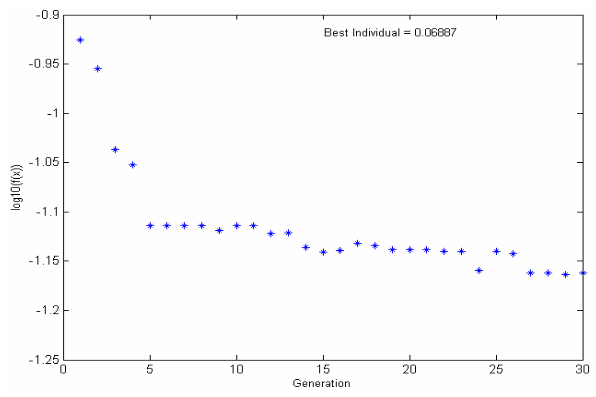


Fig. 9. Evolution of the GA for the FLC optimization

Fig 10 shows the control result of plant 2 using the optimized FLC obtained with the GA.

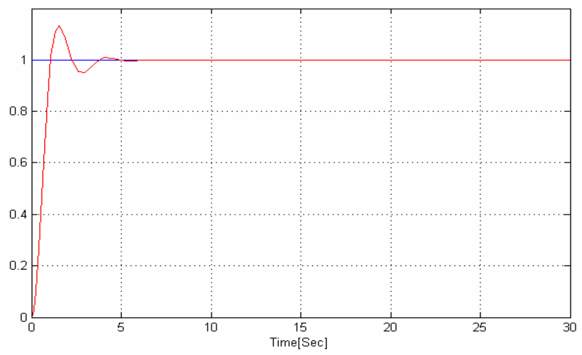


Fig. 10. Closed-loop response of plant 2 with the optimized FLC

4.4 Plant 2 Using the PSO

Table 6 presents the main results obtained with the PSO method of Plant 2 showing in row five the best result.

Table 6. Results of the FLC Obtained by PSO for Plant 2

No	Swarm	Max Iter	C1	C2	Inertia	Time exec	Average error
1	200	70	1	1	1	0:04:59	0.14286
2	200	70	0.5	0.5	1	0:07:11	0.12527
3	200	70	0.25	0.25	1	0:10:21	0.14508
4	200	70	0.15	0.15	1	0:16:25	0.15090
5	**200**	**70**	**0.05**	**0.05**	**1**	**0:17:56**	**0.13431**
6	200	70	0.005	0.005	1	0:21:18	0.14513

Fig 11 shows the membership functions of input 1 and input 2 obtained by the PSO and Fig 12 shows the particles behavior of the PSO giving the best FLC for controlling the plant 2.

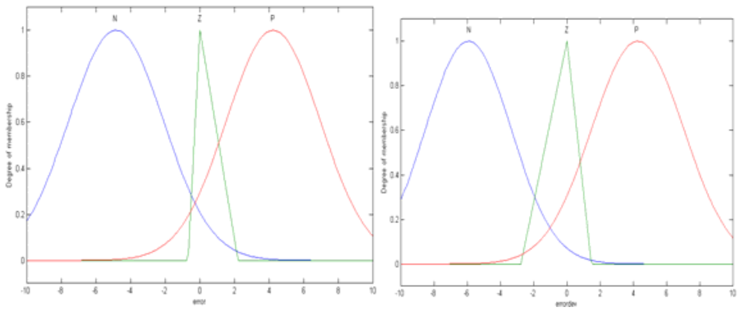


Fig. 11. Input 1 and input 2 membership functions of the optimized FLC

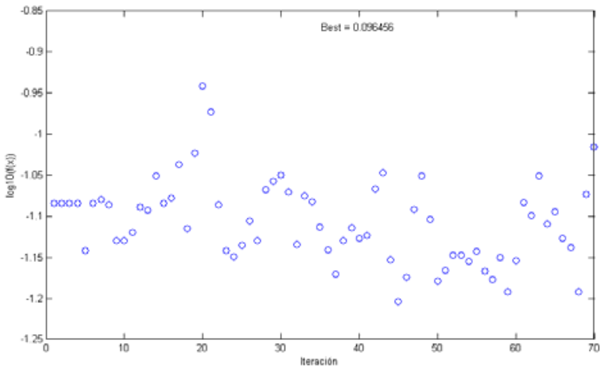


Fig. 12. Particles behavior of PSO

Fig 13 shows the control result of plant 2 using the optimized FLC obtained with the PSO.

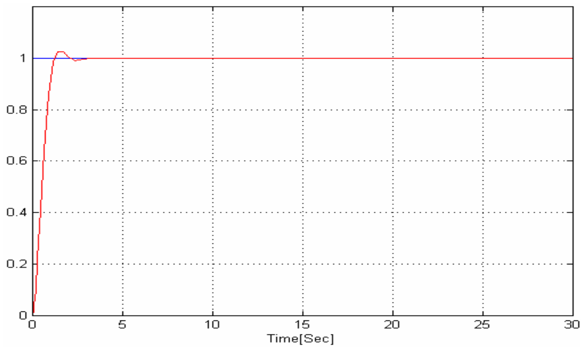


Fig. 13. Closed-loop response of plant 2 with the optimized FLC

5 Conclusion

We described in this paper the application of bio-inspired methods to design optimized fuzzy logic controllers using genetic algorithms and particle swarm optimization. To test these optimized FLC's we use different systems. In particular we presented results of a genetic algorithm and PSO using two benchmark problems with different level of complexity. The main result shows that the FLC obtained by GA and PSO gets stability in less than 10 seconds. On the other hand, the FLC's obtained by PSO are better than the FLC's obtained by GA because the PSO is less time consuming in the process and achieves lower overshoot in one of the plants, the plots of the results shows this difference.

We have achieved satisfactory results with genetic algorithms and PSO; the next step is to solve the problem using Type-2 FLC in a perturbed environment and considering multiple objective optimization to obtain better results. Moreover, we will extend the results for nonlinear systems.

References

1. Engelbretht, A.P.: Fundamentals of Computational Swarm Intelligence, pp. 5–129. John Wiley & Sons Ltd., England (2005)
2. Melin, P., Castillo, O.: A New Method for Adaptive Control of Non-Linear Plants Using Type-2 Fuzzy Logic and Neural Networks. International Journal of General Systems 1563-5104 33(2), 289–304 (2004)
3. Sepúlveda, R., Montiel, O., Castillo, O., Melin, P.: Optimizing the MFs in Type-2 Fuzzy Logic Controllers. Using the Human Evolutionary Model International Review of Automatic Control (IREACO) Theory and Applications 3(1), 1–10 (2010)
4. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan, pp. 39–43 (1995); Lu, J.-G.: Title of paper with only the first word capitalized. J. Name Stand. Abbrev (in press)
5. Fogel, D.B.: An introduction to simulated evolutionary optimization. IEEE Transactions on Neural Networks 5(1), 3–14 (1994)
6. Angeline, P.J.: Using Selection to Improve Particle Swarm Optimization. In: Proceedings 1998 IEEE World Congress on Computational Intelligence, Anchorage, Alaska, IEEE, Los Alamitos (1998)
7. Kennedy, J., Mendes, R.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
8. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceeding of IEEE Conference on Evolutionary Computation, pp. 1671–1676 (2002)
9. Angeline, P.J.: Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Differences. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 601–610. Springer, Heidelberg (1998)
10. Alcalá, R., Alcalá-Fdez, J., Herrera, F.: A proposal for the Genetic Lateral Tuning of Linguistic Fuzzy Systems and its Interaction with Rule Selection. IEEE Transactions on Fuzzy Systems 15(4), 616–635 (2007)

11. Alcalá, R., Gacto, M.J., Herrera, F., Alcalá-Fdez, J.: A Multi-objective Genetic Algorithm for Tuning and Rule Selection to Obtain Accurate and Compact Linguistic Fuzzy Rule-Based Systems. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 15(5), 539–557 (2007)
12. Casillas, J., Cordon, O., del Jesús, M.J., Herrera, F.: Genetic Tuning of Fuzzy Rule Deep Structures Preserving Interpretability and its Interaction with Fuzzy Rule Set Reduction. IEEE Transaction on Fuzzy Systems 13(1), 13–29 (2005)
13. Cordon, O., Gomide, F., Herrera, F., Hoffmann, F., Magdalena, L.: Ten Years of Genetic Fuzzy Systems: Current Framework and New trends. Fuzzy Sets and Systems 141(1), 5–31 (2004)
14. Chi, Z., Yan, H., Pham, T.: Fuzzy Algorithms: With Applications to Image Processing and Pattern recognition. World Scientific, Singapore (1996)
15. Driankov, D., Hellendoorn, H., Reinfrank, M.: An Introduction to Fuzzy Control. Springer, Berlin (1993)
16. Fukao, T., Nakagawa, H., Adachi, N.: Adaptive Tracking Control of a NonHolonomic Mobile Robot. IEEE Trans. On Robotics and Automation 16(5), 609–615 (2000)
17. Liang, Q., Mendel, J.M.: Interval Type-2 Fuzzy Logic Systems: Theory and Design. IEEE Trans. on Fuzzy Systems 8(5), 535–550 (2000)
18. Lee, T.H., Leung, F.H.F., Tam, P.K.S.: Position Control for Wheeled Mobile Robot Using a Fuzzy Controller, pp. 525–528. IEEE, Los Alamitos (1999)
19. Pedrycz, W. (ed.): Fuzzy Modelling: Paradigms and Practice. Kluwer Academic Press, Dordrecht (1996)
20. Martinez, R., Castillo, O., Aguilar, L.T., Rodriguez, A.: Evolutionary Optimization of type-2 Fuzzy Systems Applied to Linear Plants. To appear in System, Man, and Cybernetic Conference (2009)
21. Martinez, R., Castillo, O., Aguilar, L.T.: Intelligent Control For A Perturbed Autonomous Wheeled Mobile Robot Using Type-2 Fuzzy Logic and Genetic Algorithms. Journal of Automation, Mobile Robotics & Intelligent Systems 2 (2008); Wilkinson, J.P.: Nonlinear resonant circuit devices (Patent style), U.S. Patent 3 624 12 (July 16, 1990)

FPGA Implementation of Fuzzy System with Parametric Membership Functions and Parametric Conjunctions

Prometeo Cortés Antonio[1], Ildar Batyrshin[2], Heron Molina Lozano[1], Luis A. Villa Vargas[1], and Imre Rudas[3]

[1] National Polytechnic Institute, Mexico
acorteo@hotmail.com, hmolina@cic.ipn.mx, lvilla@cic.ipn.mx
[2] Mexican Petroleum Institute, Mexico
batyr1@gmail.com
[3] Obudu University, Hungria
rudas@uni-obuda.hu

Abstract. A method of FPGA implementation of fuzzy system with parametric membership functions and conjunctions is proposed. The implemented system is based on a Sugeno fuzzy model with two input variables. Fuzzy sets in the premises of the rules are given by parametric triangular membership functions and conjunction operations are defined by parametric (*p*)-monotone sum of basic t-norms. The paper presents the hardware design of a 8-bit configurable fuzzy system, implemented on the DE2 development board from Altera using VHDL language.

Keywords: FPGA, fuzzy system, conjunction, Altera, VHDL.

1 Introduction

The great popularity of fuzzy systems in solving everyday problems [1,2] has created the need in hardware implementation of highly reconfigurable fuzzy systems that can be easy adopted to various applications or to change of environment where fuzzy system is operated. Such reconfigurable fuzzy systems can be developed on two levels: on the level of the fuzzy model and on the level of hardware implementation. This paper presents a method of hardware implementation of fuzzy systems that reconfigurable on both levels. On the level of fuzzy model we consider fuzzy systems with parametric membership functions and parametric operations. A parameterization of membership functions is a common approach to construction of fuzzy systems [3]. A parameterization of operations does not so common but also used in fuzzy modeling [4-11]. But parameterization of both fuzzy sets and fuzzy operations is a sufficiently new approach in fuzzy modeling [12]. Such parameterization of fuzzy systems gives possibility to construct highly reconfigurable fuzzy systems with high adaptive possibilities. On the level of hardware we consider FPGA (Field Programmable Gate Array) implementation of fuzzy systems, i.e. an easily reprogrammable integrated circuit [13-31] that can be adopted to the change of parameters of fuzzy system and moreover to the change of the structure of fuzzy system.

Usually in fuzzy systems as conjunction operation they are used *min* or *product* operations. As parametric conjunction operations in fuzzy models it is possible to use parametric t-norms [4] or parametric fuzzy conjunctions introduced in [5,6] and having more simple form than t-norm. But both types of parametric conjunctions are complicated for hardware implementation. FPGA implementation of fuzzy systems is considered in many works [13-25]. Parametric conjunctions sufficiently simple for hardware implementation have been introduced in [22,25]. These conjunctions use basic t-norms together with parametric functions called generators. More simple parametric conjunctions suitable for hardware implementation have been introduced in [33]. These conjunctions use only basic t-norms together with some parameter p without generator functions. FPGA implementation of these operations was proposed in [34]. In our paper we extend results obtained in [34] and we show how to use these parametric operations in Sugeno fuzzy systems with parametric membership functions implemented in FPGA.

We consider Sugeno model with two inputs when each input have three linguistic values represented by parametric triangular membership values and the fuzzy system contains 9 rules. Our Sugeno fuzzy model contains parametric operation AND represented by (p)-monotone sum of basic t-norms [33]. FPGA implementation of these operations were proposed in [34] for this reason and due to limitation of the paper size we give here only very short description of hardware implementation of these operations with the reference on the original paper [34] for details.

The paper has the following structure. In the section 2 we discuss Sugeno fuzzy models that we consider in this work. The (p)-monotone sum of basic t-norms is discussed in Section 3. Section 4 presents logic diagrams of modules used in FPGA implementation of Sugeno fuzzy system with parameterized membership functions and operations. Section 5 contains discussion of simulation results. Section 6 contains conclusions and directions of future work.

2 Sugeno Fuzzy Model

Sugeno fuzzy models with two inputs consist on the following rules [3]:

$$R_i\text{: } If\ x\ is\ A_i\ AND\ y\ is\ B_i\ then\ z = f_i(x,y),$$

where x,y are input variables, A_i and B_i are fuzzy sets defined on domains X and Y of x and y respectively, z is the output of the rule R_i and f_i is a real valued function. The details of calculation of the output values of Sugeno models can be found in [3]. Fuzzy sets A_i, B_i are defined by their membership functions $A_i\colon X\rightarrow L$, $B_i\colon Y\rightarrow L$ where L is a set of membership values. In traditional fuzzy systems it is used the set of membership values $L= [0,1]$. In digital representation of membership values with m bits as in [33] we use the set of membership values $L= \{0,1,2,\ldots, 2^m\text{-}1\}$ with maximal value 2^m-1 denoted as I. This value will represent the full membership corresponding to the value 1 in traditional set of membership values [0,1]. For example, I= 15 if m= 4 and I= 255 if m= 8. Many concepts of fuzzy systems have straightforward extension on digital case when we replace the set of membership values [0,1] by $L= \{0,1,2,\ldots, 2^m\text{-}1\}$ and maximal membership value 1 by I = 2^m-1.

In our simulations we use *m*= 8 bits for representation of membership values and hence I= 255. Input variables *x* and *y* have three fuzzy values {X_S, X_M, X_L} and {Y_S, Y_M, Y_L} defined as shown in Fug. 1 by parameter values P_x and P_y respectively. These fuzzy values can be considered as formalizations of linguistic values *SMALL, MIDDLE, LARGE* of variables *x* and *y* respectively. For simplicity of fuzzy system implementation we use also *m*= 8 bits for representation of domains *X* and *Y* such that *X* = *Y* = *L*. Generally it can be used more bits for such representation but in any case we can consider such *x* and *y* as normalized inputs of fuzzy system. The fuzzy sets X_S, X_M, X_L and Y_S, Y_M, Y_L are used as fuzzy sets A_i and B_i in rules R_i and the system contains 9 rules corresponding to all possible combinations of fuzzy values A_i,B_i.

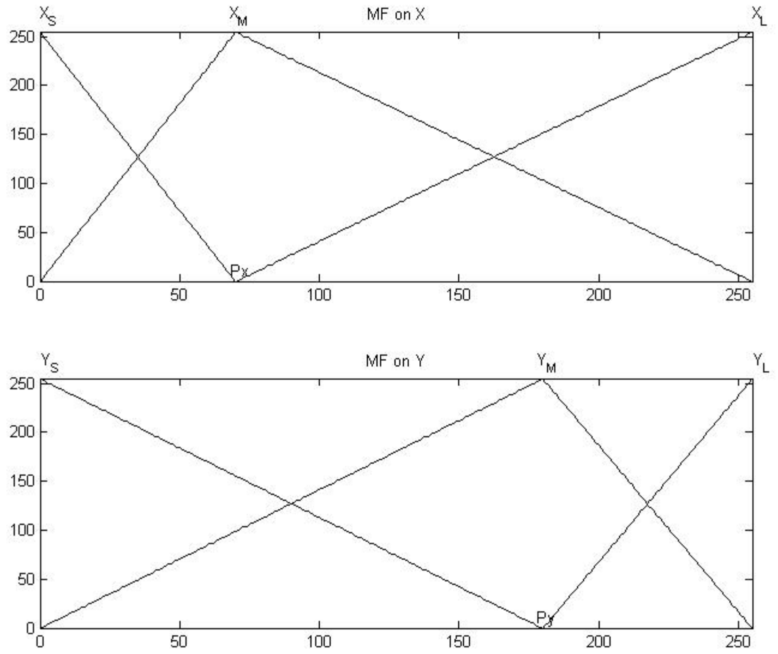


Fig. 1. Membership functions of input variables of fuzzy system depending on parameters P_x and P_y

The functions $f_i(x,y)$ are defined as follows:

$$f_i(x,y) = (a_i*x+b_i*y+c_i)/3,$$

where the parameters a_i,b_i,c_i take values in *L*. The functions f_i are constructed to take values in *L* by means of operation * that can be considered as normalized multiplication of two values from *L*. For any two values *p, q* from *L* this operation is defined as follows:

$$p*q = floor(pq/(I+1)),$$

where pq is the usual multiplication of p and q and the operation $floor(d)$ rounds the element d to the nearest integer less than or equal to d. We use such type of normalization of digital values because the usual multiplication of two digital values and the normalization of the result can be simply implemented in the DE2 development board of Altera using VHDL language [30]. The result $s=pq$ of the multiplication will take value in extended set of digital values defined by $2m$ bits and the normalization operation $floor(s/(I+1))$ can be simply implemented by cutting m right bits from the result of multiplication s.

The output of the Sugeno model for given input values x and y is calculating as follows [3]. For each rule R_i it is calculated membership values $\mu_i(x)$ of x in X_i and $\mu_i(y)$ in Y_i and the firing value of the rule $w_i= T(\mu_i(x),\mu_i(y))$ where T is a fuzzy conjunction used as connective AND. The fuzzy conjunctions used in this work will be considered in the following section. The output of the system is calculated as follows:

$$z = (w_1z_1+w_2z_2+\ldots+w_9z_9)/(w_1+w_2+\ldots+w_9),$$

where $z_i =f_i(x,y)$ is the output of the rule R_i.

In Sugeno fuzzy model we use parametric conjunction operations AND considered in the following Section.

3 Parametric Conjunctions

Fuzzy conjunction operation is a function $T:L\times L \rightarrow L$ satisfying on L conditions:

$$T(x,I) = x, \qquad T(I,y) = y, \qquad \text{(boundary conditions)}$$

$$T(x,y) \le T(u,v), \qquad \text{if } x \le u,\ y \le v. \qquad \text{(monotonicity)}$$

Commutative and associative conjunctions are called t-norms [32]. Usually in fuzzy systems as conjunction operation they are used the simplest conjunctions such as *min* o *product* operations. As parametric conjunction operations in fuzzy models it is possible to use parametric t-norms [4] or parametric fuzzy conjunctions introduced in [5,6] and having more simple form. But both types of parametric conjunctions are complicated for hardware implementation. Here, as a parametric conjunction we use (p)-monotone sum of basic t-norms that have simple FPGA implementation [33,34].

We consider here the following basic t-norms:

$$T_M(x,y) = min\{x,y\}, \qquad \text{(minimum)}$$

$$T_L(x,y)= max\{x+y - I,\ 0\}, \qquad \text{(Lukasiewicz } t\text{-norm)}$$

$$T_D(x,y) = \begin{cases} x, & \text{if } y = I \\ y, & \text{if } x = I \\ 0, & \text{if } x, y < I \end{cases}. \qquad \text{(drastic t-norm)}$$

It can be shown that any fuzzy conjunction T satisfies the following inequalities:

$$T_D(x,y) \le T(x,y) \le T_M(x,y).$$

We will say that $T_1 \le T_2$ if $T_1(x,y) \le T_2(x,y)$ for all x,y from L. For example, we have: $T_D \le T_L \le T_M$.

Suppose $a,b \in L$ and $a \le b$. The set of all elements $c \in L$ such that $a \le c \le b$ will be denoted as $[a,b]$. Suppose $p \in \{0,1,\dots, 2^m-2\}$ is a parameter. (p)-monotone sum of basic t-norms is defined as follows [33]. Define a partition of L on two sets: $X_1 = [0,p]$ and $X_2 = [p+1,I]$. These sets define a partition of $L\times L$ on four sections: $D_{ij}=X_i \times X_j$, $i,j \in \{1,2\}$ as shown in Fig. 2. Select a sequence of fuzzy conjunctions (T_{11}, T_{21}, T_{12}, T_{22}) ordered as follows: $T_{11} \le T_{12} \le T_{22}$, $T_{11} \le T_{21} \le T_{22}$. Define a function T on $L\times L$ by $T(x,y) = T_{ij}(x,y)$ if $(x,y) \in D_{ij}$, $i,j \in \{1,2\}$. This function T is called a (p)-monotone sum of fuzzy conjunctions T_{ij}, $i,j \in \{1,2\}$ and it will be a fuzzy conjunction [33]. It is clear that fuzzy conjunction T can be defined as follows:

$$T(x,y) = \begin{cases} T_{11}(x,y), & if \quad x \le p, \quad y \le p \\ T_{21}(x,y), & if \quad x > p, \quad y \le p \\ T_{12}(x,y), & if \quad x \le p, \quad y > p \\ T_{22}(x,y), & if \quad x > p, \quad y > p \end{cases}.$$

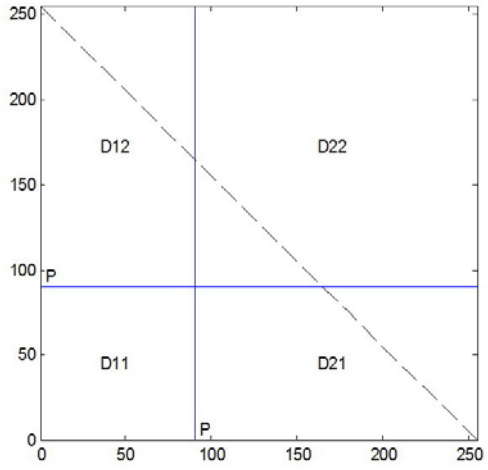


Fig. 2. Partition of $L\times L$ on sections $D_{ij}=X_i\times X_j$ defined by parameter value p

As conjunctions (T_{11}, T_{21}, T_{12}, T_{22}) in the definition of (p)-monotone sum we will use basic t-norms T_D, T_L, T_M. For example, (p)-monotone sum defined by basic t-norms (T_D, T_L, T_L, T_M) will be denoted as conjunction T_{DLLM} and defined as follows:

$$T_{DLLM}(x,y) = \begin{cases} T_D(x,y), & if \quad x \le p, \quad y \le p \\ T_L(x,y), & if \quad x > p, \quad y \le p \\ T_L(x,y), & if \quad x \le p, \quad y > p \\ T_M(x,y), & if \quad x > p, \quad y > p \end{cases}.$$

Fig. 3 depicts on the left the locations of basic t-norms used in the construction of T_{DLLM} in sections D_{11}, D_{21}, D_{12}, D_{22}. On the right it is shown the shape of this digital fuzzy conjunction when the membership scale L is defined by m= 4 bits with parameter value $p = 9$.

(p)-monotone sum will be commutative if $T_{21} = T_{12}$. By means of basic t-norms it can be constructed 7 different non-trivial (p)-monotone sums defined by the following sequences of basic t-norms: *DDDL, DDDM, DLLL, DLLM, DMMM, LLLM, LMMM.* All of them have been implemented in [34] in FPGA.

In the following section we will propose the method of FPGA implementation of these operations in Sugeno model.

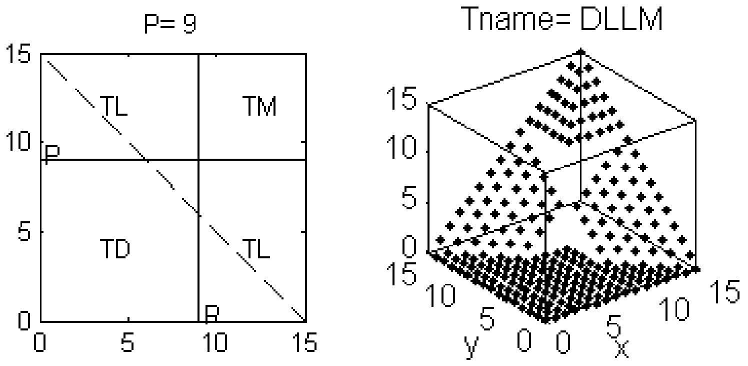


Fig. 3. Parametric conjunction T_{DLLM} obtained by (p)-monotone sum of basic t-norms: TD- drastic, TL- Lukasiewicz and TM– minimum t-norms

4 FPGA Implementation of Sugeno Model

Figures 4-6 depict logic diagrams implementing fuzzy sets of input variables. Fig. 7 depicts logic diagram of comparator of inputs for triangular fuzzy set.

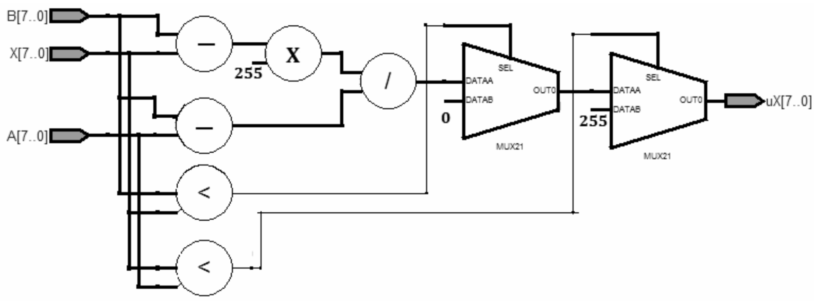


Fig. 4. Logic diagram for fuzzy set X_S

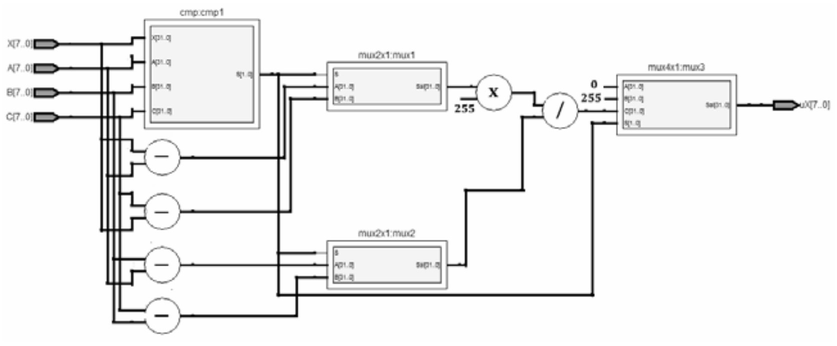


Fig. 5. Logic diagram for fuzzy set X_M with one multiplier and one divider

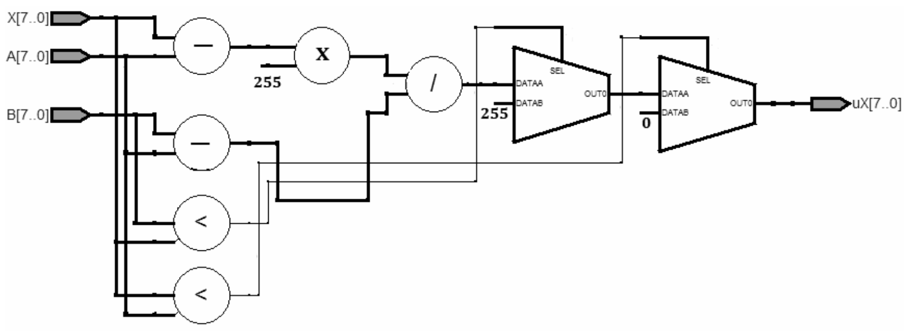


Fig. 6. Logic diagram for fuzzy set X_L

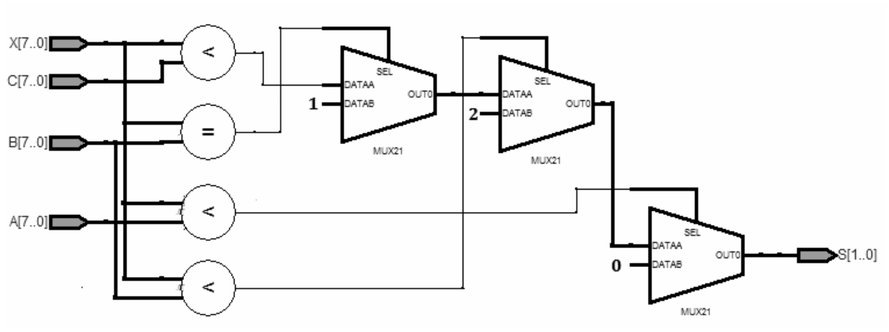


Fig. 7. Logic diagram of comparator of inputs

Logic diagrams of parametric conjunctions given by (p)-monotone sum of basic t-norms have been developed in [34]. Due to limitations in paper size here we present only some components of resulting diagrams. Figures 8-10 depict logic diagrams of basic t-norms. Fig. 11 depicts resulting logic diagram integrating 7 commutative (p)-monotone sums of basic t-norms, see [34].

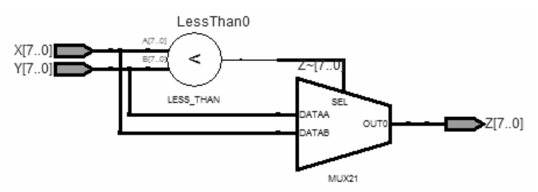


Fig. 8. Logic diagram of t-norm T_M

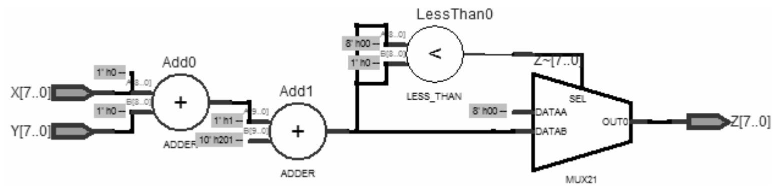


Fig. 9. Logic diagram of t-norm T_L

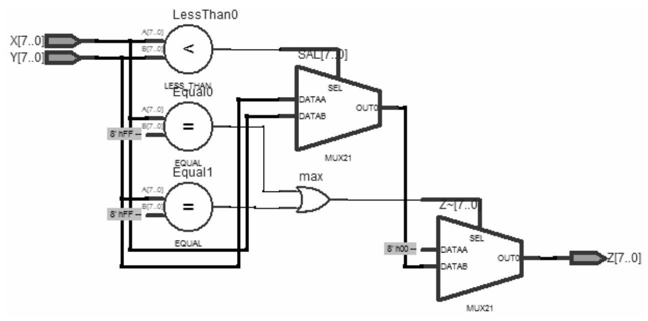


Fig. 10. Logic diagram of t-norm T_D

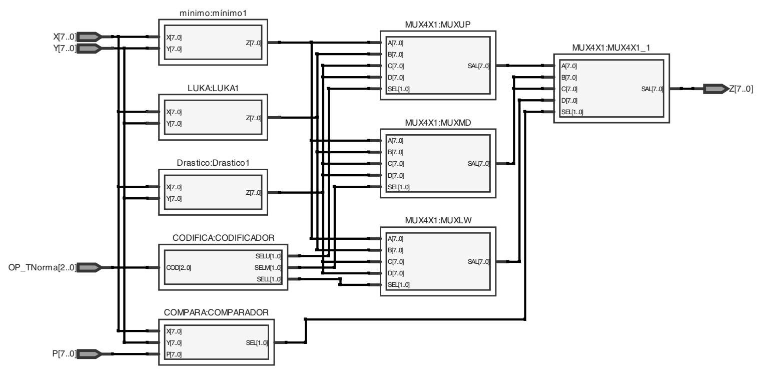


Fig. 11. Logic diagram integrating all commutative (p)-monotone sums of basic t-norms [34]

Figures 12-15 depict logic diagrams used in calculation of output of rules.

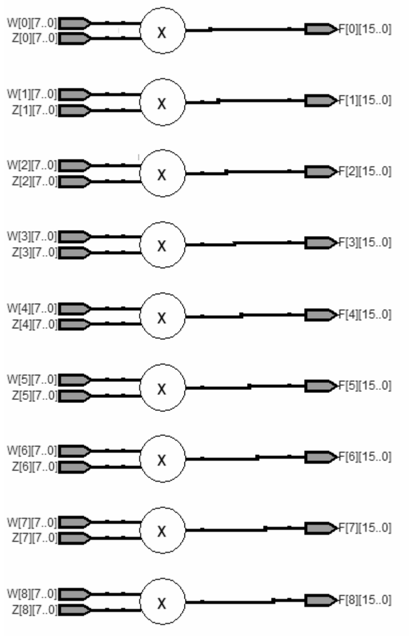


Fig. 12. Calculation of outputs of rules weighted by firing values of rules

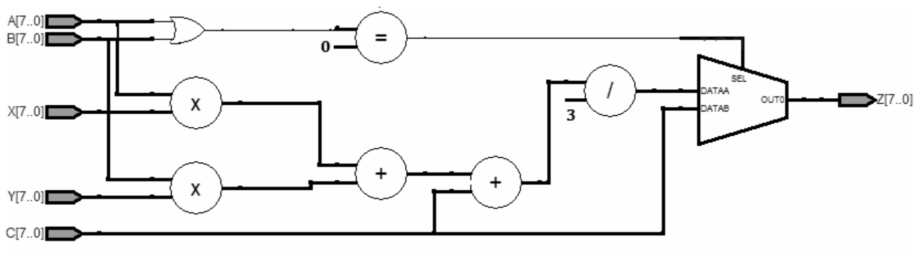


Fig. 13. Logic diagram of the output of rules

The parameters a_i, b_i, c_i are stored in a memory as shown in Fig. 14.

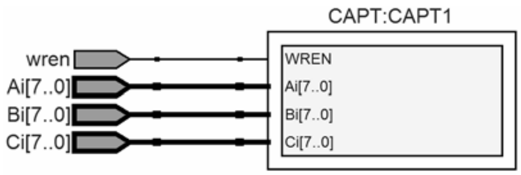


Fig. 14. Memory that stores the values of the parameters of the outputs

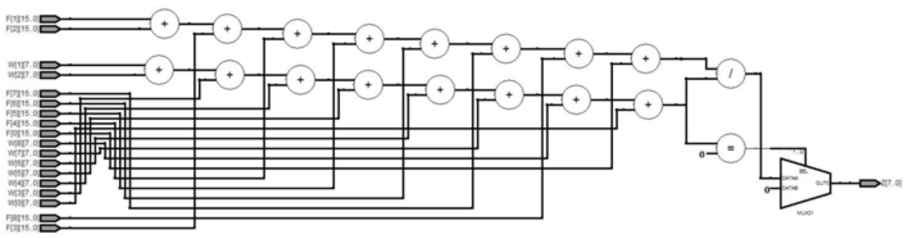


Fig. 15. Logic diagram of aggregation of outputs of fuzzy rules

5 Simulation and Discussion of Results

The developed approach was used for hardware implementation of Sugeno fuzzy systems with 2 input variables *X* and *Y*. Each input variable has 3 possible fuzzy values as shown in Fig. 1. Fuzzy system contains 9 rules corresponding to all combinations of the input fuzzy values. We have done simulation of Sugeno fuzzy models for all 7 parametric conjunctions considered above. Fig. 16 depicts obtained surfaces for parameter values P_x=70, P_y=180 and P=30.

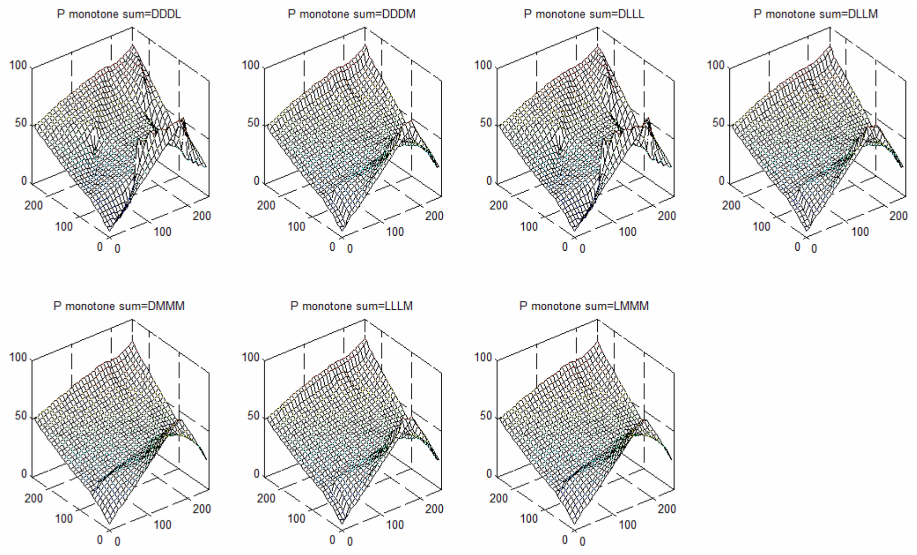


Fig. 16. Output surfaces of Sugeno fuzzy systems for all types of parametric conjunctions

The parameters *a,b,c* of linear functions in consequents of 9 rules of Sugeno fuzzy system were defined as follows:

a=[100 40 70 50 30 200 200 50 40],
b=[20 50 80 30 50 40 30 30 200],
c=[20 60 90 200 60 10 40 60 30].

Note that Sugeno fuzzy systems with parametric conjunctions considered in this paper generalize Sugeno fuzzy systems with traditional conjunction operations such as *min, drastic and Lucasiewicz* conjunctions because these basic conjunctions can be obtained from parametric conjunctions when parameter P takes values 2^m-2 or 0.

6 Conclusions

This paper presents a method of hardware implementation of highly reconfigurable fuzzy system. This system is reconfigurable on two levels: on the level of the model and on the level of the hardware implementation. This system uses Sugeno fuzzy model with parametric membership functions and parametric operations that gives possibility to adjust both membership functions and operations used in the system. The system is implemented on the DE2 development board from Altera that gives possibility to easy reconfigure hardware if underlying fuzzy model will be reconfigured. Such high reconfigurability of the system gives possibility to easy adopt the system to various application and to change of environment where fuzzy system can be applied. The obtained results can be extended on Sugeno fuzzy models with other types of parametric operations.

Acknowledgments. This work was supported in part by the Bilateral CONACYT-NKTH project No. I0110/127/08 Mod. Ord. 38/08, by ICyTDF funding, award No. PICCT08-22, and by matching funding by IPN, award No. SIP/DF/2007/143.

References

1. Terano, T., Asai, K., Sugeno, M.: Applied Fuzzy Systems. Academic Press Professional, San Diego (1994)
2. Yen, J., Langari, R., Zadeh, L.A.: Industrial Applications of Fuzzy Logic and Intelligent Systems. IEEE Press, NJ (1995)
3. Jang, J.-S.R., Sun, C.T., Mizutani, E.: Neuro-Fuzzy and Soft Computing. In: A Computational Approach to Learning and Machine Intelligence, Prentice-Hall International, Englewood Cliffs (1997)
4. Cervinka, O.: Automatic Tuning of Parametric T-norms and T-conorms in Fuzzy Modeling. In: 7th IFSA World Congress, Prague, pp. 416–421 (1997)
5. Batyrshin, I., Kaynak, O., Rudas, I.: Generalized Conjunction and Disjunction Operations for Fuzzy Control. In: 6th European Congress on Intelligent Techniques & Soft Computing, EUFIT 1998, Aachen, Germany, vol. 1, pp. 52–57 (1998)
6. Batyrshin, I., Kaynak, O.: Parametric Classes of Generalized Conjunction and Disjunction Operations for Fuzzy Modeling. IEEE Transactions Fuzzy Systems 7, 586–596 (1999)
7. Bikbulatov, A., Batyrshin, I.: Tuning of Operations in Fuzzy Models by Neural Nets. In: 7th Zittau Fuzzy Colloquium, Zittau, Germany, pp. 142–147 (1999)
8. Eskil, M.T., Efe, M.O., Kaynak, O.: T-norm Adaptation in Fuzzy Logic Systems Using Genetic Algorithms. In: IEEE Intern. Symposium on Industrial Electronics, ISIE 1999, Bled, Slovenia, vol. 1, pp. 398–402 (1999)
9. Batyrshin, I., Kaynak, O., Rudas, I.: Fuzzy Modeling Based on Generalized Conjunction Operations. IEEE Transactions Fuzzy Systems 10, 678–683 (2002)

10. Koprinkova-Hristova, P.D.: Fuzzy Operations' Parameters Versus Membership Functions' Parameters Influence on Fuzzy Control Systems Properties. In: 2nd IEEE Int. Conf. on Intelligent Systems, pp. 219–224 (2004)
11. Alcalá-Fdez, J., Herrera, F., Márquez, F., Peregrín, A.: Increasing Fuzzy Rules Cooperation Based on Evolutionary Adaptive Inference Systems. Intern. J. Intelligent Systems 22, 1035–1064 (2007)
12. Aras, A.C., Kaynak, O., Batyrshin, I.Z.: A Comparison of Fuzzy Methods for Modeling. In: 34th Annual Conf. of the IEEE Industrial Electronics Society, IECON 2008, Orlando, pp. 43–48 (2008)
13. McKenna, M., Wilamowski, B.M.: Implementing a Fuzzy System on a Field Programmable Gate Array. In: Intern. Joint Conf. Neural Networks, IJCNN 2001, Washington, DC, vol. 1, pp. 189–194 (2001)
14. Sánchez-Solano, S., Senhadji, R., Cabrera, A., Baturone, I., Jiménez, C.J., Barriga, A.: Prototyping of Fuzzy Logic–Based Controllers Using Standard FPGA Development Boards. In: 13th IEEE International Workshop on Rapid System Prototyping, Darmstadt, RSP 2002, pp. 25–32 (2002)
15. Mermoud, G., Upegui, A., Peña, C., Sanchez, E.: A Dynamically-Reconfigurable FPGA Platform for Evolving Fuzzy Systems. In: Cabestany, J., Prieto, A.G., Sandoval, F. (eds.) IWANN 2005. LNCS, vol. 3512, pp. 572–581. Springer, Heidelberg (2005)
16. Di Stefano, A., Giaconia, C.: An FPGA-Based Adaptive Fuzzy Coprocessor. In: Cabestany, J., Prieto, A.G., Sandoval, F. (eds.) IWANN 2005. LNCS, vol. 3512, pp. 590–597. Springer, Heidelberg (2005)
17. Deliparaschos, K.M., Nenedakis, F.I., Tzafestas, S.G.: Design and Implementation of a Fast Digital Fuzzy Logic Controller Using FPGA Technology. J. Intelligent Robotic Systems 45, 77–96 (2006)
18. Sanchez-Solano, S., Cabrera, A.J., Baturone, I., Moreno-Velo, F.J., Brox, M.: FPGA Implementation of Embedded Fuzzy Controllers for Robotic Applications. IEEE Trans. Indust. Electronics 54, 1937–1945 (2007)
19. Govindasamy, K., Neeli, S., Wilamowski, B.M.: Fuzzy System with Increased Accuracy Suitable for FPGA Implementation. In: INES 2008, Intelligent Engineering Systems Conf., pp. 133–138 (2008)
20. Montiel, O., Olivas, J., Sepúlveda, R., Castillo, O.: Development of an Embedded Simple Tuned Fuzzy Controller. In: Zurada, J.M., Yen, G.G., Wang, J. (eds.) WCCI 2008. LNCS, vol. 5050, pp. 555–561. Springer, Heidelberg (2008)
21. Montiel, O., Maldonado, Y., Sepúlveda, R., Castillo, O.: Simple Tuned Fuzzy Controller Embedded into an FPGA. In: NAFIPS 2008 Conference Proceedings, New York (2008)
22. Hemandez Zavala, A., Batyrshin, I.Z., Rudas, I.J., Villa Vargas, L., Camacho Nieto, O.: Parametric Operations for Digital Hardware Implementation of Fuzzy Systems. In: Hernández Aguirre, A., et al. (eds.) MICAI 2009. LNCS (LNAI), vol. 5845, pp. 432–443. Springer, Heidelberg (2009)
23. Huang, Y.J., Wu, B.C., Chen, C.Y., Chang, C.H., Kuo, T.C.: Solar Tracking Fuzzy Control System Design Using FPGA. In: World Congress on Engineering, London, vol. 1, pp. 340–344 (2009)
24. Obaid, Z.A., Sulaiman, N., Hamidon, M.N.: FPGA-Based Implementation of Digital Logic Design Using Altera DE2 Board. Int. J. Computer Science and Network Security 9, 186–193 (2009)
25. Rudas, I.J., Batyrshin, I.Z., Hernández Zavala, A., Camacho Nieto, O., Villa Vargas, L.: Digital Fuzzy Parametric Conjunctions for Hardware Implementation of Fuzzy Systems. In: IEEE 7th Int. Conf. Computational Cybernetics ICCC2009, pp. 157–166. Palma de Mallorca, Spain (2009)

26. Kilts, S.: Advanced FPGA Design. Architecture, Implementation, and Optimization. John Wiley & Sons, New Jersey (2007)
27. Brown, S., Vranesic, Z.: Fundamentals of Digital Logic with VHDL Design, 2nd edn. Mc Graw Hill, New York (2005)
28. Palnitkar, S.: Verilog HDL: A Guide to Digital Design and Synthesis, 2nd edn. (2008)
29. Cyclone II Device Handbook, Vol. 1, Altera (2008), `http://www.altera.com/literature/hb/cyc2/cyc2_cii5v1.pdf`
30. DE2 Development and Education Board User Manual. User Manual. Versión 1.4. Altera (2006), `ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf`
31. Xilinx Inc., System Generator User Guide, `http://www.xilinx.com`
32. Klement, E.P., Mesiar, R., Pap, E.: Triangular Norms. Kluwer, Dordrecht (2000)
33. Batyrshin, I.Z., Rudas, I.J., Panova, A.: On Generation of Digital Fuzzy Parametric Conjunctions. SCI, vol. 243, pp. 79–89. Springer, Heidelberg (2009)
34. Cortés Antonio, P., Batyrshin, I., Rudas, I., Panova, A., Villa Vargas, L.: FPGA Implementation of (p)-Monotone Sum of Basic t-norms. In: IEEE World Congress on Computational Intelligence, FUZZ-IEEE WCCI 2010, Barcelona, Spain, pp. 1491–1497 (2010)

Fuzzy Logic Hardware Implementation for Pneumatic Control of One DOF Pneumatic Robot

Juan-Manuel Ramos-Arreguin, Emmanuel Guillen-Garcia, Sandra Canchola-Magdaleno, Jesus-Carlos Pedraza-Ortega, Efren Gorrostieta-Hurtado, Marco-Antonio Aceves-Fernández, and Carlos-Alberto Ramos-Arreguin

Universidad Autónoma de Querétaro, Fac. De Informática
Campus Juriquilla, Av. de las Ciencias S/N,
Juriquilla, Delegacion Santa Rosa Jauregui, CP 76230
Querétaro, México
juan.ramos@uaq.mx, sandra.canchola@uaq.mx, marco.aceves@uaq.mx

Abstract. The pneumatic actuators can be a useful way to control the position of a manipulator robot, instead of an electrical actuator. The major problem with pneumatic actuators is the compressibility of the air, due to the fact that the mathematical model is a system formed by a set of highly non-linear equations then a simple PID control is not enough to control the robot position, and fuzzy logic is a good option. This work is focused on the hardware implementation of Fuzzy logic algorithm into FPGA system. This paper also presents a methodology to implement a pneumatic control using fuzzy logic, into a FPGA device, which is the main contribution of this work. The air flow is controlled with a pulse width modulation, applied to the pneumatic electro- valve, with 25 ms of period.

Keywords: FPGA, Hardware, Fuzzy Logic, pneumatic, Robot.

1 Introduction

Fuzzy logic is widely used to control many industrial systems, especially for systems with a no linear behavior, where a PID control is not enough to get satisfactory results. However, the use of fuzzy logic algorithms implemented in hardware, has been used in different ways. Fuzzy logic algorithms and its hardware implementation have been developed for several kinds of control, such as Barriga and Sánchez-Solano [1] who develop an automatic synthesis of fuzzy logic controllers, and its implementation [2]. Later, in 1998, an XFuzzy software is reported [3]; this software is used to develop fuzzy algorithms and its implementation. A digital fuzzy controller has been implemented in 1999, with SISO (single-input-single-output) and MIMO (multiple-input-multiple-output) models [4]. In 2004, a hardware implementation of fuzzy system type 2 has been developed [5]. A fuzzy controller is implemented for electric vehicle, for fuzzy systems with dynamic reconfiguration, implemented with an FPGA; the methodology used is co-evolutionary cooperation; the objective is to increase the computing speed of the system [6] [7]. The popularity of fuzzy logic controllers for industrial applications is increasing with time, especially when the mathematical models are not available or inaccurate; for that a hardware implementation of the fuzzy algorithm is necessary with FPGA technology [8]. However, fuzzy logic implementation can be used for traffic control [9].

About hardware implementation of fuzzy controller on FPGAs, several works have been developed. In 1994, a fuzzy implementation is developed into a microcontroller for embedded control solutions [10a]. In 2008 is presented a flexible architecture that allows to implement an embedded nonlinear fuzzy controller into FPGA [10b] [10c], using a Simple Tuning Algorithm. Also, a generic algorithm is used into a Fuzzy system for FPGA implementation [10d].

In robotic field, the pneumatic actuators has been used previously in [10], [11], [12], [13], where several kind of controllers are used for position control, including PID, fuzzy logic and neural networks, with practical results comparisons. This paper is focused into fuzzy logic hardware implementation for position control of a pneumatic robot with one degree of freedom, as shown in figure 1, where $\boldsymbol{X}$ the rod displacement and $\boldsymbol{\theta}$ is the arm position with vertical reference.

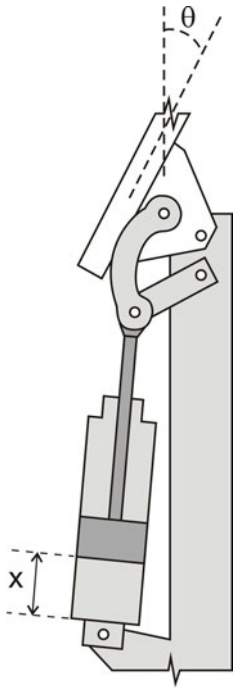


Fig. 1. Pneumatic robot with one degree of freedom

The contribution of this work is the process to control an electro-valve, for rod displacement, such as shown next. The control is applied for PWM signal, applied to the electro-valve, using a hardware implementation of a fuzzy algorithm into a FPGA.

2 The Pneumatic System

The pneumatic system used for the pneumatic robot is shown in figure 2. The digital signals EV_1 and EV_2 are used to control the air flow in the pneumatic cylinder. To control the air flow a 5 ports 4 way and 3 position electro-valve with closed center is used.

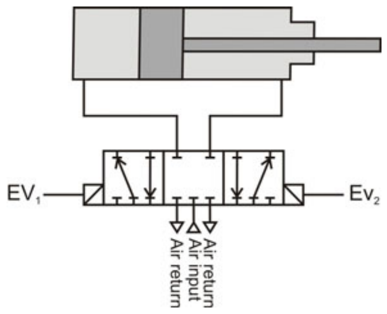


Fig. 2. Pneumatic system used for on one degree of freedom robot

The cylinder used in this work, is the same used in [10], with chambers on both sides as figure 3 shown, and its mathematical model is explained in next section.

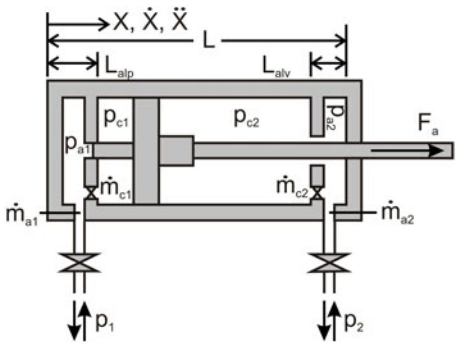


Fig. 3. A pneumatic cylinder diagrams, with internal variables of pressure and air mass flow

2.1 Mathematical Model

The mathematical model, called Thermo-Mechanical simplified model in [10] and corresponding to the pneumatic cylinder (it is shown in figure 3), is defined in (1) to (10). The mathematical model is divided in several sections, according with the rod displacement defined by X, as shown in figure 1.

For the interval $0 \leq X \leq L$:

$$\dot{X} = \frac{d}{dt} X \tag{1}$$

$$D\dot{X} = \frac{d^2}{dt^2} X \tag{2}$$

For the interval $0 \le X \le L_{alp}$:

$$\dot{P}_{a1} = g_{21}(X)\left(\dot{m}_{a1} - \dot{m}_{c1} - 9.176\times10^{-10} P_{a1} DX\right)\times10^{8} \tag{3}$$

$$\dot{P}_{c1} = g_{31}(X)\left(\dot{m}_{c1} - 3.608\times10^{-8} P_{c1} DX\right)\times10^{6} \tag{4}$$

For the interval $L_{alp} < X \le L$:

$$\dot{P}_{a1} = g_{22}(X)\left(\dot{m}_{a1} - 3.7\times10^{-8} P_{a1} DX\right)\times10^{11} \tag{5}$$

$$\dot{P}_{c1} = g_{32}(X)\left(\dot{m}_{c1} - 3.7\times10^{-8} P_{c1} DX\right)\times10^{11} \tag{6}$$

For the interval $0 \le X \le (L\text{-}L_{alv})$:

$$\dot{P}_{c2} = g_{41}(X)\left(\dot{m}_{c2} + 3.469\times10^{-8} P_{c2} DX\right)\times10^{11} \tag{7}$$

$$\dot{P}_{a2} = g_{51}(X)\left(\dot{m}_{a2} + 3.469\times10^{-8} P_{a2} DX\right)\times10^{11} \tag{8}$$

For the interval $(L - L_{alv}) < X \le L$:

$$\dot{P}_{c2} = g_{42}(X)\left(\dot{m}_{c2} + 3.352\times10^{-8} X_4 X_6\right)\times10^{13} \tag{9}$$

$$\dot{P}_{a2} = g_{52}(X)\begin{bmatrix} 9.983\times10^{3}\left(\dot{m}_{a2} - \dot{m}_{c2}\right) + \\ 1.168\times10^{-5} X_5 X_6 \end{bmatrix}\times10^{4} \tag{10}$$

Due to the non-linear behavior of the air, and the non-linear mathematical model, the system behavior is complex. For this reason a fuzzy logic algorithm is used in this work.

2.2 Air Mass Flow Control

With the use of PWM method, the air flow can be controlled, and the rod can be changed its position inside the pneumatic cylinder. The electro-valve is a SY5320-6LZ-01, manufactured by SMC Company, with 19 s time delay. A PWM cycle with its timing data is shown in figure 4; this PWM signal is applied on the electro-valve to control the rod speed.

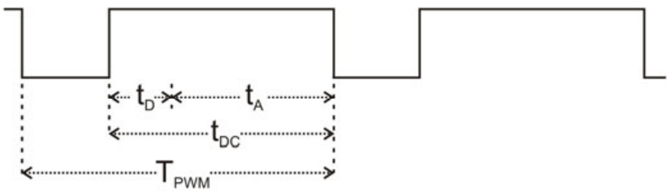


Fig. 4. Timing diagram of PWM signal

To control the displacement direction of the rod, digital signals EV_1 and EV_2 are used. If the rod has to spin backwards to the cylinder, signal EV_2 must be set HIGH and signal EV_1 is set LOW. However, if the rod has to go out the cylinder, signals EV_1 and EV_2 are set HIGH and LOW, respectively. The rod speed depends directly of t_A time, and this time is set by the fuzzy logic algorithm. Table 1 shows the meaning of each variable involved in figure 4. The fuzzy Logic is develop to adjust the t_A time, and control the air flow.

The time of duty cycle is divided in 256 parts, depending of the speed desired by the system. The Fuzzy Logic algorithm gives the output needed by the controller.

Table 1. Variables involved in PWM timing diagram

Variable	Description	Unit
T_{PWM}	Cycle of the PWM	Seconds
t_{DC}	Duty cycle time	Seconds
t_D	Delay time	Seconds
t_A	High time applied	Seconds

3 The Fuzzy Algorithm

The duty cycle of PWM signal (t_{DC}) is set by the fuzzy algorithm. If t_{DC} is lower than t_D, the rod does not move by any means. Conversely, if t_{DC} is equal or greater than t_D, the rod may move in any way, either fast or slow.

Figure 5 shows a block diagram of the system to control the arm position, controlling the angle θ. With the sign of the error signal, the direction of rod movement is defined, and the electro-valve gets the PWM signal from the PWM generator, and the air flow is used into the pneumatic cylinder. Due to the robot topology, the most important variable is the output angle value. Therefore, the path planning for the robot and the angle speed control are not a problem.

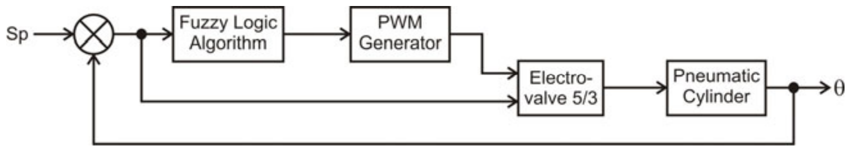


Fig. 5. Block diagram to control the rod displacement ***X***

In figure 5, first step to consider is to compute the error. The error is the input of the fuzzy logic algorithm, where the output is the duty cycle value, needed for the PWM Generator block. This block deliver the PWM signal to the electro-valve, and joint with the error, the rod movement is generated in the right direction.

The fuzzy logic algorithm is designed with Matlab software. The algorithm considers just the error (E) as the input, and the output is the duty cycle for the PWM generator. Figure 6 and 7 show the membership functions for the input and the output variables. The direction of rod movement is defined by the error, defined by (11).

$$E = Sp - X . \tag{11}$$

Where:

E = Error
Sp = Set point
X = Actual position

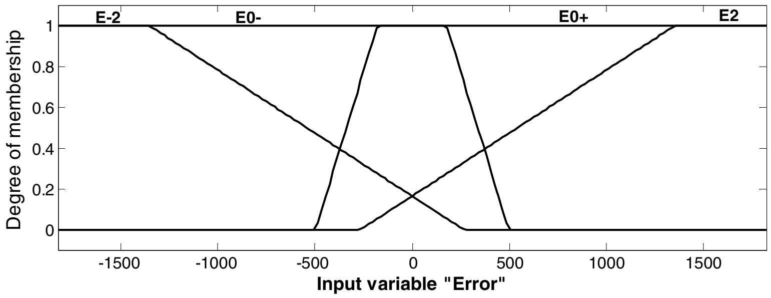


Fig. 6. Membership functions for the input (Error)

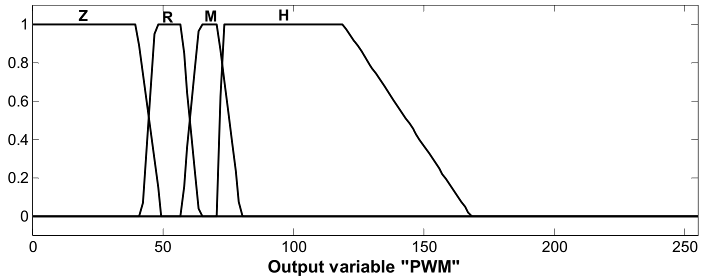


Fig. 7. Membership functions for the output (DC)

The rules used in this case are shown next.

```
Rule 1: if Error is E-2 then DC is H
Rule 2: if Error is E2 then DC is H
Rule 3: if Error is E0+ then DC is Z
Rule 4: if Error is E0- then DC is Z
```

The actual position is sensed by an optical encoder. The interval of movement of the arm is 0 to 170 degrees, represented by 0 to 1820 counts of the encoder. Therefore, the resolution of movement measure is 0.0934° by each count of the encoder. The output interval is 0 to 255, this value represent the duty cycle value for PWM generator, and give us a good resolution for the flow control.

The method used for rules evaluation is Mamdani, and the defuzzification method is centroid. By experimentation, the minimum value of DC to get a rod displacement is 47, and the movement is faster with 75. Therefore, the output range for DC is from 45 to 75, as figure 8 shows.

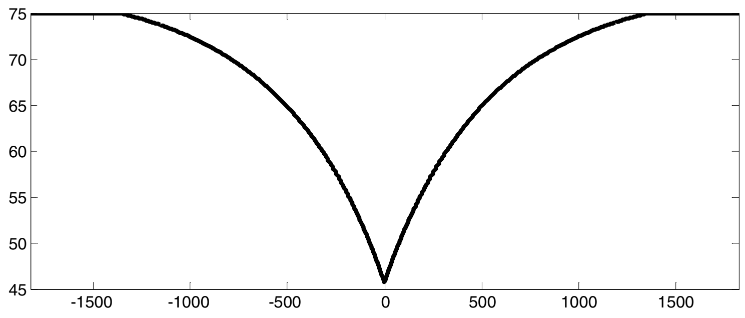


Fig. 8. Surface of the fuzzy rules (Error input versus DC output)

The hardware implementation of the fuzzy logic algorithm is explained in the next section.

4 Hardware Implementation

The fuzzy algorithm hardware implementation has been developed in several ways, as proposed in [10c], where a hardware realization is presented, including fuzzification and inference engine. Their proposal to improve the fuzzification is based on the arithmetic calculation method. Also, this realization reduces the hardware cost by mean of reducing its complexity. However, when the hardware cost is reduced, the time solution is increased. The realization proposed in this work, has the main advantage in the computing speed, due that the defuzzification result is obtained in just one clock cycle. The LUT shown in figure 9 contain the result of the fuzzy rules and the output value. The process to obtain the LUT is explained on 4.1.

The hardware implementation is developed using a Xilinx board Spartan-3 Starter Kit, with an FPGA device XC3S200, with 200K gates. A hardware description language is needed to implement the fuzzy algorithm, and VHDL is the used description language. The description block diagram is shown in figure 9.

The Counter module takes the pulses from the optical encoder, and signals B and SEG are used to show the arm position with 4 seven segments display. The counter output is the signal C with 12 bit wide. The signal C is the input of the Error module, where (1) is computed; the output is the signal R. The signal R is connected to the look-up table (LUT) module, to get the output for the PWM duty cycle; also, the more significant bit of signal R is used to determine if the rod goes outward or goes inward. If the error is positive, then PWM signal goes out through the EV_1 signal; when the error is negative, the PWM signal is assigned to EV_2. The PWM module gets the value of Duty Cycle with M signal, and the rod direction with S signal.

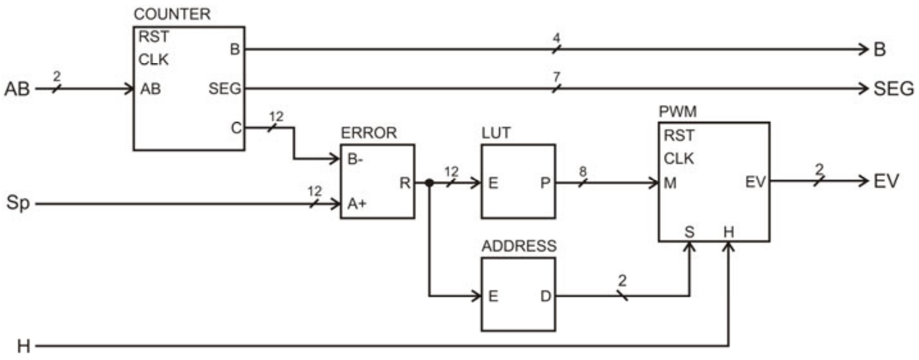


Fig. 9. Block diagram of the fuzzy logic hardware implementation

Next, the hardware implementation of fuzzy algorithm is described.

4.1 Implementation of Fuzzy Logic

First, the fuzzy logic is designed in MATLAB, then the LUT module shown in figure 8 is created. To create this module, a process of module generation is used as shown in figure 10, where n is the bit resolution used for the fuzzy system. In this case n=8. With n>8, the PWM change is not detected by.

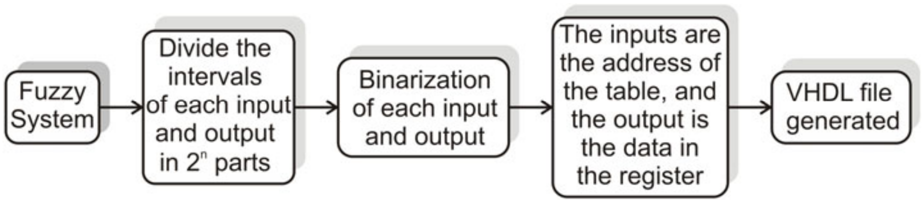

Fuzzy System
Divide the intervals of each input and output in 2ⁿ parts
Binarization of each input and output
The inputs are the address of the table, and the output is the data in the register
VHDL file generated

Fig. 10. Process for look-up table generation

The Matlab code used to get a digital Fuzzy Logic Algorithm as LUT, is shown next.

```
% Open VHDL file
 fid = fopen('fuzzy_PWM.vhd','wt');

 %Description header
 fprintf(fid,'library IEEE;\n');
 fprintf(fid,'use IEEE.std_logic_1164.all;\n');
 fprintf(fid,'\n');
 fprintf(fid,'entity fuzzy_PWM is\n');
 fprintf(fid,'   port(\n');
 %Coefficient of Error as index
 fprintf(fid,'      E : in  std_logic_vector(%d downto 0);\n',m-1);
 %Defuzzification coefficient DC
 fprintf(fid,'      DC: out std_logic_vector(%d downto 0)\n',k-1);
 fprintf(fid,'      );\n');
 fprintf(fid,'   end fuzzy_PWM;\n');
 fprintf(fid,'\n');
 fprintf(fid,'architecture LUT of fuzzy_PWM is\n');
 fprintf(fid,'begin\n');
 fprintf(fid,'   process(E)\n');
 fprintf(fid,'   begin\n');
 fprintf(fid,'      case E is\n');
 %Automatic table generation
 for i=1:n
     fprintf(fid,'         when "');
     %Index binarization
     for j=1:m
         fprintf(fid,'%d',x(i,j));
     end;
     fprintf(fid,'" => P <= "');
     %Coefficient binarization
     for j=1:k
         fprintf(fid,'%d',DAT(i,j));
     end;
     %Decimal values
     fprintf(fid,'"; -- Index %d   Coefficient %d\n',i-1,out(i));
 end;
 %End VHDL file
 fprintf(fid,'         when others => null;\n');
 fprintf(fid,'      end case;\n');
 fprintf(fid,'   end process;\n');
 fprintf(fid,'end LUT;\n');

 %Close file
 fclose(fid);
```

The X signal has the digital information of the Error input, and DAT has the digital information of DC output. This MATLAB algorithm generates the LUT module of figure 9. The out of the module LUT is connected to the PWM Generator Module, which is described next.

4.2 PWM Module

A descriptive block diagram of PWM module is shown in figure 11. The timing values used for PWM signal for figure 4 are shown in table 2.

Table 2. Timing values assigned for table 1

Variable	Value	Unit
T_{PWM}	25	miliseconds
t_{DC}	0 a 25	miliseconds
t_D	19	miliseconds
t_A	0 a 7	miliseconds

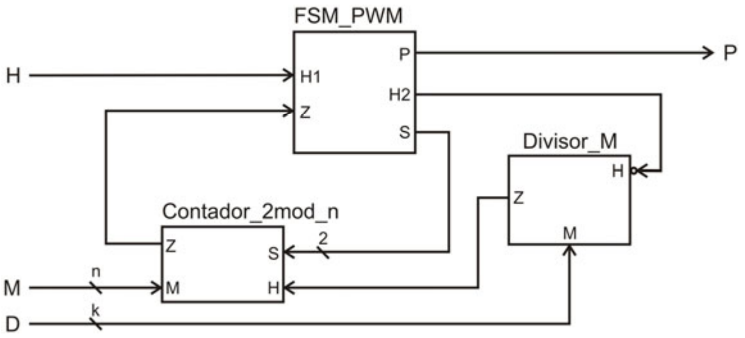


Fig. 11. PWM hardware description

Where T_{CLK} is the clock period. Therefore, the clock period is T_{CLK} = 50 ns. D value computed is D=1250000, in hexadecimal is \$1312D0, with 21 bit, according with (12).

$$D = T_{PWM} / T_{CLK} . \quad (12)$$

5 Results

The Spartan-3 board and power interface to control the electro-valve are shown in figure 12; also pneumatic manipulator with one degree of freedom is shown in the same figure. The air pressure to work with the cylinder is 0.4 MPa, and the step response is shown in figure 14.

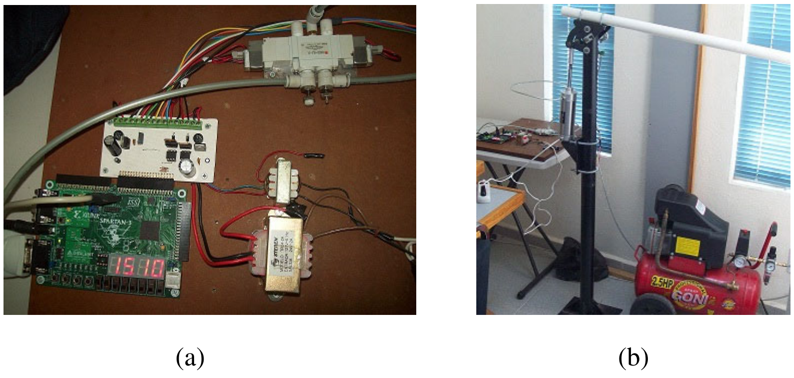


Fig. 12. System of the pneumatic robot with one degree of freedom. (a) Spartan-3 kit, power interface and electro-valve used with the pneumatic manipulator. (b) A pneumatic manipulator with one degree of freedom.

The implementation of the system into FPGA let us to verify the system behavior. Figure 13 shows the test, with different values of set point.

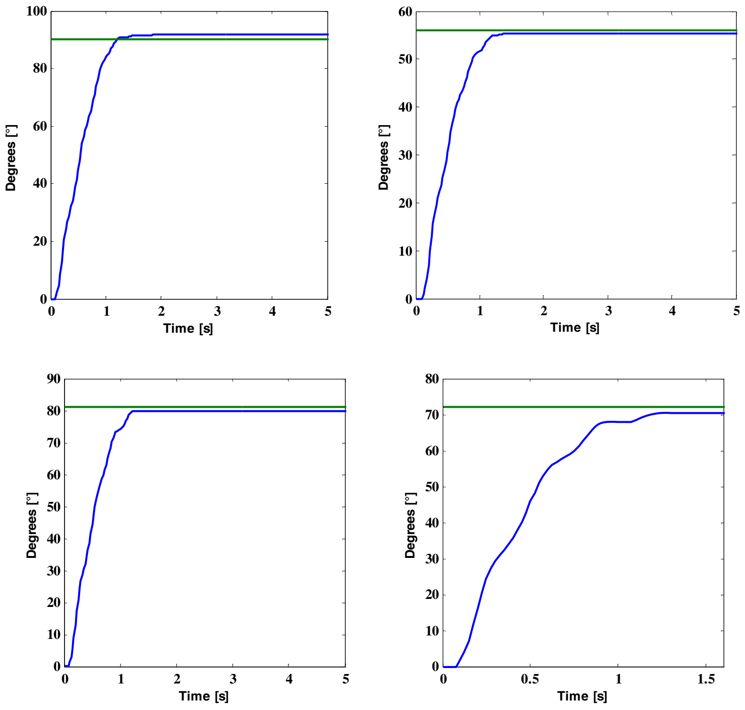


Fig. 13. Test of the system response, with different set point

In figure 13, the flat line is the set point, the error in all cases are around 1°, which is good for this test of implementation.

6 Conclusions

This work is using a Fuzzy Logic algorithm to control the arm position, and the hardware implementation is very important, because the system has not any dependency of a PC. In other works developed, the hardware implementation of Fuzzy Logic has been oriented with electrical motors, systems without a mathematical model, but works with hardware implementation applied to pneumatic cylinder is not reported. The major advantage of this realization is the computing speed, due that the fuzzy algorithm is computed in just one clock cycle. This let us to make other process in parallel, such as RS232 communication, data acquisition, etc. To continue with this work, a planning trajectory will be developed, also, a second degree should be added to the actual robot.

As future work, a multivariable control have to be implemented, with more degree of freedom for the robot, considering more complex path planning and the pressure control into the chambers of pneumatic cylinder.

References

1. Barriga, A., Sánchez-Solano, S., Jiménez, C.J., Gálan, D., López, D.R.: Automatic Synthesis of Fuzzy Logic Controllers. Mathware & Soft Computing 3, 425–434 (1996)
2. Lago, E., Hinojosa, M.A., Jiménez, C.J., Barriga, A., Sánchez-Solano, S.: FPGA Implementation of Fuzzy Controllers. In: XII Conference on Design of Circuits and Integrated Systems (DCIS 1997), pp. 715–720 (1997)
3. López, D.R., Jiménez, C.J., Baturone, I., Barriga, A., Sánchez-Solano, S.: Xfuzzy: A Design Environment for Fuzzy Systems. In: Seventh IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 1998), Anchorage, Alaska, pp. 1060–1065 (1998)
4. Patyra, M.J., Grantner, J.L.: Hardware implementations of digital fuzzy logic controllers. Information Sciences: an International Journal 113, 19–54 (1999)
5. Melgarejo, M.A., Peña-Reyes, C.A.: Hardware Architecture and FPGA Implementation of a Type-2 Fuzzy System. In: Proceedings of the 14th ACM Great Lakes symposium on VLSI, pp. 458–461 (2004)
6. Mermoud, G., Upegui, A., Peña, C.A., Sanchez, E.: A Dynamically-Reconfigurable FPGA Platform for Evolving Fuzzy Systems. In: Cabestany, J., Prieto, A.G., Sandoval, F. (eds.) IWANN 2005. LNCS, vol. 3512, pp. 572–581. Springer, Heidelberg (2005)
7. Poorani, S., Urmila-Priya, T.V.S., Kumar, K.U., Renganarayanan, S.: Fpga Based Fuzzy Logic Controller for Electric Vehicle. Journal of The Institution of Engineers 45(5), 1–14 (2005)
8. Deliparaschos, K.M., Nenedakis, F.I., Tzafestas, S.G.: Design and Implementation of a Fast Digital Fuzzy Logic Controller Using FPGA Technology. Journal of Intelligent and Robotic Systems 45, 77–96 (2006)
9. Karakuzu, C., Demirci, O.: Fuzzy Logic Based Smart Traffic Light Simulator Design and Hardware Implementation. Applied Soft Computing 10(1), 66–73 (2010)

10a. Bannatyne, R.: Development of fuzzy logic in embedded control. Sensor Review 14(3), 11–14 (1994)

10b. Montiel, O., Olivas, J., Sepúlveda, R., Castillo, O.: Development of an Embedded Simple Tuned Fuzzy Controller. In: Zurada, J.M., Yen, G.G., Wang, J. (eds.) WCCI 2008. LNCS, vol. 5050, pp. 555–561. Springer, Heidelberg (2008)

10c. Montiel, O., Maldonado, Y., Sepulveda, R., Castillo, O.: Simple tuned fuzzy controller embedded into an FPGA. Annual Meeting of the North American Fuzzy Information Processing Society 19(22), 1–6 (2008)

10. Ramos, J.M., Gorrostieta, E., Pedraza, J.C., Romero, R.J., Ramírez, B.: Pneumatic Cylinder Control for a Flexible Manipulator Robot. In: 12th IEEE International Conference on Methods and Models in Automation and Robotics, pp. 637–641 (2006)
11. Gorrostieta, E., Ramos, J.M., Pedraza, J.C.: Fuzzy and Neuronal Control to Flexible Manipulator. International Journal of Factory Automation, Robotics and Soft Computing, 155–160 (April 2007)
12. Ramos-Arreguin, J.M., Pedraza-Ortega, J.C., Gorrostieta-Hurtado, E., Romero-Troncoso, R.J., Vargas-Soto, J.E., Hernandez-Hernandez, F.: Pneumatic Fuzzy Controller Simulation vs Practical Results for Flexible Manipulator. Automation and Robotics, 191–200 (2008)
13. Ramos-Arreguin, J.M., Pedraza-Ortega, J.C., Gorrostieta-Hurtado, E., Romero-Troncoso, R.J.: Artificial Intelligence Applied into Flexible Manipulator. In: Seventh Mexican International Conference on Artificial Intelligence, pp. 339-345 (2008)

Author Index