Studies in Computational Intelligence 343

Kurosh Madani
António Dourado Correia
Agostinho Rosa
Joaquim Filipe (Eds.)

Computational Intelligence

Revised and Selected Papers of the International Joint Conference IJCCI 2009 held in Funchal-Madeira, Portugal, October 2009

Springer

Kurosh Madani, António Dourado Correia, Agostinho Rosa, and Joaquim Filipe (Eds.)

Computational Intelligence

Studies in Computational Intelligence, Volume 343

Editor-in-Chief

Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Kurosh Madani, António Dourado Correia,
Agostinho Rosa, and Joaquim Filipe (Eds.)

Computational Intelligence

Springer

Prof. Kurosh Madani
University PARIS-EST Creteil (UPEC)
Images, Signals and Intelligence Systems Laboratory
LISSI EA 3956
Paris 12
France
E-mail: madani@univ-paris12.fr

Prof. António Dourado Correia
University of Coimbra
Departamento de Engenharia Informatica
Polo II - Pinhal de Marrocos
3030 Coimbra
Portugal
E-mail: dourado@eden.dei.uc.pt

Prof. Agostinho Rosa
Instituto Superior Tecnico IST
Systems and Robotics Institute
Evolutionary Systems and Biomedical Engineering Lab
Av. Rovisco Pais
1049-001 Lisboa
Portugal
E-mail: acrosa@laseeb.org

Prof. Joaquim Filipe
Polytechnic Institute of Setúbal / INSTICC
2910-595 Setubal
Portugal
E-mail: jfilipe@insticc.org

ISBN 978-3-642-20205-6 e-ISBN 978-3-642-20206-3

DOI 10.1007/978-3-642-20206-3

Studies in Computational Intelligence ISSN 1860-949X

Library of Congress Control Number: 2011925687

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

If “Artificial Intelligence” (AI), defined by John McCarthy (who coined the term in 1956) as “the science and engineering of making intelligent machines”, is considered since five decades as being a key branch of Computer Science, “Computational Intelligence” (CI) appears today as an outstanding offshoot of this key branch. As an alternative to symbolic AI, the CI rather relies on the combination of heuristic algorithms with formal optimization techniques such as in neural networks, fuzzy systems, evolutionary computation, and their synergic combinations, embracing as well techniques which deal with Swarm Intelligence, Fractals (and Chaos Theory), Artificial Immune Systems, Digital Signal Processing such as Wavelets, etc.. Associated with soft computing, connectionist systems and cybernetics, CI combines elements of learning, adaptation, evolution and fuzziness (using Fuzzy Logic) to create approaches (paradigms, architectures, programs, etc.) that are capable of adapting to changing complex dynamic systems, , i.e., are to some extend intelligent. The CI field has shown a promising potential for technology and industrial world, providing a deep challenging lifting for many of the most difficult industrial and technological computational problems.

The present book includes extended and revised versions of a set of selected papers from the First International Joint Conference on Computational Intelligence (IJCCI 2009), held in Funchal - Madeira, Portugal, from 5 to 7 October, 2009.

Spotting that the field of CI, as most of highly technical and specialized research domains, is deeply divided into subfields that often fail to communicate, the purpose of IJCCI is to bring together researchers, engineers and practitioners in computational technologies, especially those related to the areas of fuzzy computation, evolutionary computation and neural computation. IJCCI is composed of three co-located Conferences, each one specialized in at least one of the aforementioned main knowledge areas. Namely:

- International Conference on Fuzzy Computation
- International Conference on Evolutionary Computation
- International Conference on Neural Computation~

The three Conferences aim to provide a major forum for scientists, engineers and practitioners interested in the study, analysis, design, modelling and implementation of systems using CI, both theoretically and in a broad range of application fields.

The International Conference on Fuzzy Computation (ICFC) encompasses the theory and application of fuzzy sets and fuzzy logic to the solution of information processing and systems-analysis problems. Bolstered by information technology developments, the extraordinary growth of Fuzzy Computation in recent years has led to major applications in fields ranging from medical diagnosis and automated learning to image understanding and systems control.

Evolutionary Computation, considered a subfield of computational intelligence focused on combinatorial optimization problems, is associated with systems that use computational models of evolutionary processes as the key elements in design and implementation, i.e. computational techniques which are inspired to some degree on the evolution of biological life in the natural world. A number of evolutionary computational models have been proposed, including evolutionary algorithms, genetic algorithms, the evolution strategy, evolutionary programming, and artificial life.

Neural Computation and artificial neural networks have seen an explosion of interest over the last decades, and are being successfully applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics, in problems of prediction, classification and control. Several promising architectures, learning strategies and algorithms have been introduced in this highly dynamic field in the last couple of decades.

IJCCI has received 231 paper submissions from more than 35 countries in all continents. 21 papers were published and presented as full papers, i.e. completed work, 35 papers reflecting work-in-progress or position papers were accepted for short presentation, and another 29 contributions were accepted for poster presentation. These numbers, leading to a “full-paper” acceptance ratio of 9% and a total oral paper presentations acceptance ratio of 24%, show the intention of preserving a high quality forum for the next editions of this conference. This book includes revised and extended versions of a strict selection of the best presented papers.

Furthermore, IJCCI 2009 included 5 plenary keynote lectures given by Janusz Kacprzyk, Jouko Lampinen, Leonid Perlovsky, Qiang Shen and Edward Tsang. We would like to express our appreciation to all of them and in particular to those who took the time to contribute with a paper to this book.

On behalf of the Conference Organizing Committee, we would like to thank all participants. First of all the authors, whose quality work is the essence of the conference and the members of the Program Committee, who helped us with their valuable expertise and diligence in reviewing the papers. As we all know, producing a post-conference book, within the high technical level exigency, requires the effort of many individuals. We wish to thank also all the members of our Organizing Committee, whose work and commitment were invaluable.

September 2010

Kurosh Madani
António Dourado
Agostinho Rosa
Joaquim Filipe

Conference Committee

Conference Co-chairs

Joaquim Filipe	Polytechnic Institute of Setúbal / INSTICC, Portugal
Janusz Kacprzyk	Systems Research Institute / Polish Academy of Sciences, Poland

Program Co-chairs

António Dourado	University of Coimbra, Portugal (ICFC)
Kurosh Madani	The University of Paris XII, France (ICNC)
Agostinho Rosa	IST, Technical University of Lisbon, Portugal (ICEC)

Organizing Committee

Patrícia Alves	INSTICC, Portugal
Sérgio Brissos	INSTICC, Portugal
Helder Coelhas	INSTICC, Portugal
Vera Coelho	INSTICC, Portugal
Andreia Costa	INSTICC, Portugal
Bruno Encarnação	INSTICC, Portugal
Carla Mota	INSTICC, Portugal
Vitor Pedrosa	INSTICC, Portugal
José Varela	INSTICC, Portugal
Pedro Varela	INSTICC, Portugal

ICFC Program Committee

Shigeo Abe, Japan
Sansanee Auephanwiriyakul, Thailand
Ulrich Bodenhofer, Austria
France Cheong, Australia
Francisco Chiclana, UK
Martine De Cock, Belgium
Bijan Davvaz, Iran, Islamic Republic of
Ioan Despi, Australia
Scott Dick, Canada
Gary Feng, Hong Kong

ICFC Auxiliary Reviewers

ICEC Program Committee

Jiri Kubalik, Czech Republic
Antonio J. Fernández Leiva, Spain
Daniele Loiacono, Italy
Manuel Lozano, Spain
Francisco Luna, Spain
Wenjian Luo, China
Evelyne Lutton, France
Rainer Malaka, Germany
Euan William McGookin, UK
Bob McKay, Korea, Republic of
Barry McMullin, Ireland
Jörn Mehnen, UK
Luiza de Macedo Mourelle, Brazil
Nysret Musliu, Austria
Schütze Oliver, Mexico
Gary Parker, USA
Shahryar Rahnamayan, Canada
Mateen Rizki, USA
Ralf Salomon, Germany
Siddhartha Shakya, UK
Konstantinos Sirlantzis, UK
P. N. Suganthan, Singapore
Jonathan Thompson, UK
Peter Tino, UK
Peter Whigham, New Zealand
Shiu Yin Yuen, China
Mengjie Zhang, New Zealand

ICEC Auxiliary Reviewer

Colin Johnson, UK

ICNC Program Committee

Waleed Abdulla, New Zealand
Shigeo Abe, Japan
Sabri Arik, Turkey
Amir Atiya, Egypt
Antonio Padua Braga, Brazil
Ivo Bukovsky, Czech Republic
Katherine Cameron, UK
Jinde Cao, China
Zheru Chi, Hong Kong
Seungjin Choi, Korea, Republic of
Chien-Hsing Chou, Taiwan
Netta Cohen, UK
José Alfredo Ferreira Costa, Brazil
Barbara Hammer, Germany
Tom Heskes, The Netherlands
Chris Hinde, UK
Akira Hirose, Japan
Tingwen Huang, Qatar
Christel Kemke, Canada
DaeEun Kim, Korea, Republic of
Edmund Lai, New Zealand
H. K. Lam, UK
Sin Wee Lee, UK
Xiaoli Li, China
Honghai Liu, UK
Bao-Liang Lu, China
Jinwen Ma, China
Kurosh Madani, France
Maciej Mazurowski, USA
Ali Minai, USA
Seiichi Ozawa, Japan
Danil Prokhorov, USA
Gerald Schaefer, UK
Jiri Sima, Czech Republic
Humberto Sossa, Mexico
Mu-Chun Su, Taiwan
Shun-Feng Su, Taiwan
Shiliang Sun, China
Johan Suykens, Belgium
Norikazu Takahashi, Japan
Ah Hwee Tan, Singapore
Brijesh Verma, Australia
Hua-Liang Wei, UK
Yingjie Yang, UK

ICNC Auxiliary Reviewers

Cyril Brom, Czech Republic	You Ji, China
Qiaona Chen, China	Petr Savicky, Czech Republic
Ya Gao, China	Roman Vaculin, Czech Republic
Rongqing Huang, China	Qingjiu Zhang, China

Invited Speakers

Janusz Kacprzyk	Systems Research Institute / Polish Academy of Sciences, Poland
Jouko Lampinen	Helsinki University of Technology, Finland
Leonid Perlovsky	Harvard University, USA
Qiang Shen	Aberystwyth University, UK
Edward Tsang	University of Essex, UK

Contents

Invited Papers

Mechanisms of the Brain and Cultures

Leonid Perlovsky

Harvard University, SEAS, Cambridge and the AF Research Laboratory
Sensors Directorate, Hanscom AFB, U.S.A.
leonid@seas.harvard.edu

Abstract. Mathematical and neural mechanisms of concepts, emotions, instincts, imaginations, intuitions are described. The paper reviews past mathematical difficulties of modeling the mind and a new mathematical theory of dynamic logic (DL) and neural modeling fields (NMF), which overcome these difficulties. DL evolves fuzzy logic into crisp one. Orders of magnitude improvement is achieved in recognition, fusion, predictions. NMF-DL is extended to a hierarchical mind system, to language and interaction of language and cognition. It follows that language, symbolic abilities, and higher cognitive functions could only evolve jointly. Models of neural mechanisms are developed for the mind hierarchy; they explain roles of the beautiful, music, sublime in the mind, cognition, and evolution of languages and cultures. DL is related to the knowledge instinct, a mechanism that drives the mind to understand the world. This instinct is more important than sex or food. Two mechanisms of the knowledge instinct, differentiation and synthesis, determine evolution of consciousness and cultures, and explain three types of cultural evolutions. The knowledge-acquiring cultures lead to science and technology, but doubt their values; traditional cultures invest limited amount of knowledge with strong emotions, values are certain but knowledge stagnates; multi-cultural societies combine knowledge and values. Different languages lead to different emotionalities and different evolutionary paths of cultures. Differences in emotional mechanisms of languages lead to different paths of cultural evolution and affect cultures no less than lexical contents.

Keywords: Language, Cognition, Emotions, Knowledge instinct, Dynamic logic, Mind, Hierarchy, Evolution of cultures.

1 Mechanisms of Language: Recent Development

Scientific approaches to the brain mechanisms of language were initiated in the 1950s by Chomsky (1965). He identified problems about language that science had to resolve, which seemed mysterious. “Poverty of stimulus” addressed the fact that the tremendous amount of knowledge needed to speak and understand language is learned by every child even without special education. Chomsky emphasized that surrounding languages do not carry enough information for a child to learn language from experience, unless specific language learning mechanisms are inborn in the human mind. This approach to language based on innate mechanisms, is called *nativism*.

Many linguists disagreed with separation between language and cognition in Chomsky's theories. Cognitive linguistics emerged in the 1970s to unify language and cognition, and explain creation of meanings. This direction emphasized that the knowledge of language is no different from the rest of cognition, and based on conceptual mechanisms. It is embodied and situated in the environment [38], [13], [18], [19], [101]. Cognitive linguistics as well as nativism has not led to a computational linguistic theory explaining how languages are learned and meanings are created. Mathematical approaches to Chomskyan and cognitive linguistics are dominated by logic, which as we discuss later is inadequate.

Evolutionary linguistics emphasized joint evolution of language and meanings. Language mechanisms are shaped by transfer from generation to generation. [35], [12]. Mathematical models in evolutionary linguistics emphasized simulation of societies of communicating agents. This approach demonstrated emergence of a compositional language [5].

2 Mechanisms of Perception and Cognition

Consider a seemingly simple experiment of object perception. Close eyes and imagine an object in front of you. The imagined image is vague, not as crisp and clear as with opened eyes. As we open eyes, the object becomes crisp and clear. It seems to occur momentarily, but actually it takes 1/5th of a second. This is a very long time for neural brain mechanisms – hundreds of thousands of neural operations. Let us also note: with opened eyes we are not conscious about initially vague imagination, we are not conscious about the entire 1/5th of a second, we are conscious only about the end of this process: crisp, clear object in front of our eyes. The explanation of this experiment has become simple after many years of research that have found out what goes on in the brain during these 1/5th of a second. Below we summarize brain mechanisms necessary for understanding this experiment.

2.1 Instincts, Emotions, Concepts

To understand this experiment we need to consider mechanisms of concepts, instincts, and emotions identified during recent decades. Concepts or mental representations are like internal models of objects and situations, stored in memory, during visual perception of an object, a concept-model of the object stored in memory projects an image (top-down signals) onto the visual cortex, which is matched there to an image projected from retina (bottom-up signals, [28]).

The concepts mechanism evolved for instinct satisfaction. We use the word instinct to denote a simple inborn, non-adaptive mechanism described in [29]. Instinct is a mechanism of internal "sensor," which measures vital body parameters, such as blood pressure, and indicate to the brain when these parameters are out of safe range (for more details see [26], [30]. We have dozens of such sensors, measuring sugar level in blood, body temperature, pressure at various parts, etc.

According to the instinctual-emotional theory [29], communicating satisfaction or dissatisfaction of instinctual needs from instinctual parts of the brain to decision making parts of the brain is performed by emotional neural signals. The word emotion refers to several neural mechanisms in the brain [6], [37], in this paper 'emotions' always refer to the mechanism connecting conceptual and instinctual brain regions.

Projection of top-down signals from a model to the visual cortex "primes" visual neurons, or makes them more receptive to matching bottom-up signals. This projection produces imagination that we perceive with closed eyes, as in the closed-opened eye experiment. Conscious perception occurs after top-down and bottom-up signals match [8]. This process of matching presented difficulties to mathematical modeling, as discussed below.

2.2 Combinatorial Complexity and Logic

Computer mechanisms of perception still cannot compete with those of kids and animals. Most algorithms and neural networks suggested since the 1950s for modeling perception and cognition, as discussed in [54], [57], [62]), faced difficulty of combinatorial complexity (CC). These CC difficulties are related to Gödelian limitations of logic [27], they are manifestations of logic incompleteness in finite systems [57]. Even approaches designed specifically to overcome logic limitations, such as fuzzy logic and neural networks, encountered logical steps in their operations and faced CC [62].

A mathematical theory of dynamic logic (DL) was proposed to overcome these difficulties [89], [49], [50], [51], [57], [62], [63], [68], [69], [74], [43]. Here we describe it conceptually. Whereas logic operates with static statements (e.g. "this is a chair," or "if A then B"), dynamic logic is a process from vague to crisp, from vague statements, decisions, plans, to crisp ones.

DL models the opened-closed eye experiment: initial states of models are vague. This experiment was recently performed with much more details using brain imaging [2]. It demonstrated that the image generated by top-down signals (imagination) is *vague-fuzzy*, similar to imagination in the closed-opened-eye experiment. Conscious perception of an object occurs when vague projections become crisp in the process of matching the crisp and clear image from the retina, and an object is consciously recognized, the entire process lasts about 160 ms.

Mechanisms of DL are necessary for perception, otherwise an organism will not be able to perceive the surroundings and will not be able to survive. The instinctual mechanism drives top-down signals to fit bottom-up signals. This mechanism is called the need for knowledge [7] or the knowledge instinct KI, [89], [52], [62], [78]. As discussed in [41] biologists considered similar mechanisms since the 1950s, without a mathematical formulation, however, its fundamental role in cognition was difficult to discern. Emotional signals of satisfaction or dissatisfaction of the KI we feel as harmony or disharmony between our knowledge-models and the world. At lower layers of the mind hierarchy, such as everyday object perception, these emotions are usually below the level of consciousness, at higher layers of abstract and general concepts this feeling of harmony or disharmony could be strong, as discussed in [63] it is a foundation of our higher mental abilities. A mathematical theory of KI is described in [62], [70].

2.3 Perception Example

A neurally inspired system implementing DL is described in [89], [57], [62], it is called Neural Modeling Fields (NMF). Figure 1 illustrates NMF-DL using an example described in [62]. It demonstrates that the described theory can find patterns below clutter at about 100 times better in terms of signal-to-clutter ratio, than previous state-of-the-art algorithms. Fig. 1b illustrates signal under clutter as available for processing. Figs. 1c through h illustrate DL process from vague-to-crisp, which finds uncertain signals under clutter without combinatorial complexity.

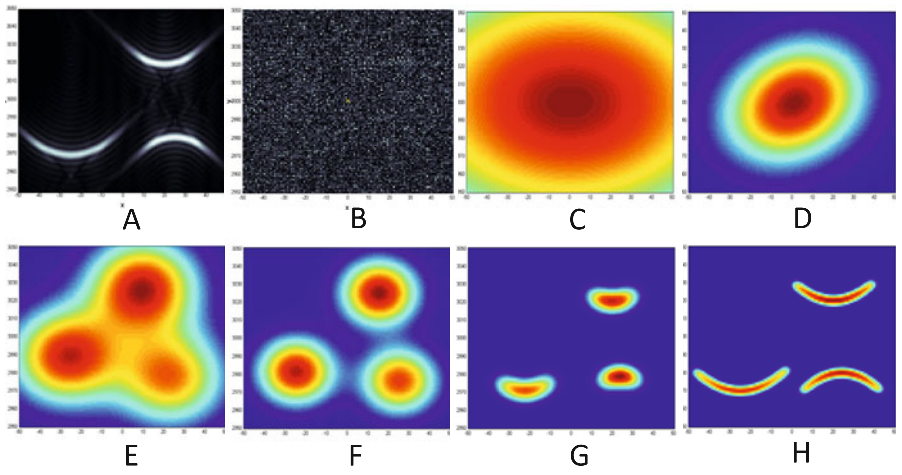


Fig. 1. Finding 'smile' and 'frown' patterns in noise, an example of dynamic logic operation: (a) true 'smile' and 'frown' patterns are shown without noise, (b) actual image available for recognition (signal is below noise, signal-to-noise ratio is between ½ and ¼), (c) an initial fuzzy blob-model, the fuzziness corresponds to uncertainty of knowledge, (d) through (h) show improved models at various iteration stages (total of 22 iterations). Between stages (d) and (e) the algorithm tried to fit the data with more than one model and decided, that it needs three blob-models to 'understand' the content of the data. There are several types of models: one uniform model describing noise (it is not shown) and a variable number of blob-models and parabolic models, which number, location, and curvature are estimated from the data. Until about stage (g) the algorithm 'thought' in terms of simple blob models, at (g) and beyond, the algorithm decided that it needs more complex parabolic models to describe the data. Iterations stopped at (h), when similarity (2) stopped increasing. This example is discussed in more details in [42].

2.4 Classical Engineering Applications of NMF-DL

This ability of dynamic logic to extract signals from strong noise and clutter was used in many applications [77]. NMF-DL algorithms improved performance over prior algorithms in several classical engineering applications by 10,000% or better. These include clustering, [89], [57]. For example when clustering with Gaussian Mixture Models (GMM), DL results in the maximum likelihood model estimation achieving information-theoretic performance limits, Cramer-Rao Bounds (CRB), [48]. Another classical application is tracking. From the NMF-DL point of view, the difference between tracking and clustering is in the models. Whereas in clustering the models are points in multidimensional feature spaces, in NMF-DL tracking models describe tracks in 2 or 3 dimensional geometric coordinate spaces. This view on tracking as clustering has been revolutionary, when first published in 1991 (see references in [54], [57], [93], [92], [94]. It led to breakthrough improvements for tracking in clutter, to maximum likelihood tracking in strong clutter achieving information-theoretic performance limits of CRB. Also DL enabled derivation of CRB for tracking in clutter. All of these have been previously considered impossible (see references in [51], [53], [57], [55], [17]. Algorithms in this area have been continuously improving since the WWII, nevertheless, popular algorithms for tracking in clutter grossly under-perform information-theoretic bounds [99], [97]. NMF-DL tracker improved performance of tracking algorithms by 10,000% [81]. When tracking in clutter, tracking (estimating track parameters) and association (deciding which data points belong to which track, or to clutter) have to be performed jointly, so called "track-before-detect," This problem is often considered NP-complete and therefore unsolvable.

Important classical engineering area is fusion of signals from multiple sensors, platforms, or data bases. In dense (or strong) clutter, detecting relevant signals or information may not be possible in a single sensor image, or in a single data base. The problem is similar to tracking, detection, tracking, and fusion have to be performed concurrently, sometimes it is called "fuse-before-track" or "fuse-before-detect," these problems are usually considered unsolvable because of CC. Similar situation exists in data mining, when mining multiple data bases, how would the algorithm know that a word or phrase in one data base is related to a telephone call in another data base, unless say, a keyword allows the algorithm to connect the relevant pieces of information. For fusion, the NMF-DL equations in the previous sections require no modifications, the data have now an additional index, indicating a sensor (or data source) and correspondingly the models have to be developed for each sensor. Data and models may include geometric measurements and classification features as well, the latter case is called feature-added fusion. In [16] NMF-DL has been applied to a complicated case of feature-added fusion using sensors on three platforms, the S/C ratio was inadequate to detect objects in a single frame, or even to track and detect using a single sensor, joint fusion, tracking, and detection of that complexity were considered previously unsolvable.

Extracting cognitive events from EEG signals could be utilized to allow quadriplegics to move a computer cursor or steer their wheelchairs with their thoughts, or those playing computer games could control actions on the screen with their thoughts. Complexity of these problems is due to uncertain character of patterns corresponding to cognitive events and high noise and clutter in EEG signals. Detecting these signals and estimating their parameters is discussed in [39]. Other engineering problems solved using DL, which were not solvable previously are described in [87], [84], [49], [50], [79], [80], [90], [91], [92], [94], [15], [85], [86].

Learning context and situations has been an unsolved problem for decades. It is a next step beyond pattern recognition both in complexity and in generality. The complexity of learning situations is due to the fact that situations are sets of objects among a huge number of objects that are not relevant to this situation, no to any other meaningful situations, and sorting through subsets of large sets encounters combinatorial complexity. Learning contexts and situations is a step toward abstract cognition beyond recognition of objects and patterns. Solving this problem required defining vagueness of contexts and situations, this vagueness is not limited to fuzzy representations of objects as in Fig.1, but also requires vagueness of context and situation contents, achieved by probabilistic models, in which contexts and situations are defined by probabilistic associations of objects and contexts [36].

3 Language and Cognition

A fundamental step both toward understanding brain mechanisms and developing future man-machine cooperative engineering systems is modeling mechanisms of language, cognition, and their interaction. All linguistic theories, as reviewed at the beginning of the paper, are formulated as logical systems, and face combinatorial complexity. This is the reason why computers do not understand human language, why Google, Yahoo, and other search engines, while being immensely useful, cause so much frustrations to their users. Extension of DL to language promises to remedy the situation [60], [64], [67], [72], [20], [21], [22], [23], [24], [36], [82], [100].

Learning phrase-models composed of words, or composition of larger chunks of text from smaller chunks of texts is similar to learning cognitive models of higher layers, say images composed of objects [36], [82]. This procedure provides a mathematical foundation for perceptual symbol systems described in [3], [83]. In particular, DL models the mathematical procedure for Barsalou “simulators” which support abilities for abstract concepts, propositions, and productivity.

Productivity describes the ability of the human mind to use finite means (words, concepts) to produce virtually infinite (combinatorially large) number of more complex structures, abstract concepts, phrases, texts. In linguistics, productivity has been associated with recursion [10], [95], [3], [33], while non-combinatorial mathematical procedures for acquisition-learning of this ability have not been previously demonstrated. This problem has been mathematically solved in [36], [82], [83], where a mathematical procedure for non-combinatorial learning is developed, which results in combinatorially powerful productivity. Recursion is implemented through the hierarchy.

Interaction between cognition and learning has been full with puzzles. Do we use phrases to label situations that we already have understood, or the other way around, do we just talk without understanding beyond words? It is obvious that different people have different cognitive and linguistic abilities and may tend to different poles in cognitive-language continuum, while most people are somewhere in the middle in using cognition to help with language, and vice versa. What are the neural mechanisms that enable this flexibility? How do we learn which words and objects come together among nearly infinite number of incorrect associations? Why kids learn language by 5 or 7, but do not think like adults? Why there are no animals with human level cognition without language and vice versa?

Little is known about neural mechanisms integrating language and cognition and no mathematical models have been ever proposed, which avoid combinatorial complexity. Here we propose a computational model that potentially can answer the above questions, and that is computationally tractable, it does not lead to combinatorial complexity. It is experimentally testable. Also it implies a relatively simple neural mechanism, which is also supported by the mechanism of mirror neurons [1], and explains why human language and human cognition are inextricably linked. It suggests that human language and cognition have evolved jointly.

3.1 Dual Model

According to [60], [62], [64], [72] integration of language and cognition is accomplished by a dual model. Every model in the human mind is not separately cognitive or linguistic, and still cognitive and linguistic contents are separate to a significant extent. Every concept-model has two parts, linguistic and cognitive.

In a newborn mind both types of models are vague empty placeholders for future cognitive and language contents. The neural connections between the two types of models are inborn, the mind never has to learn which word goes with which object. As models acquire specific contents in the process of learning, linguistic and cognitive contents are always staying properly connected.

During the first year, infants learn some objects and situations in the surrounding world: cognitive parts of some models at the layer of objects become less vague and acquire a degree of specificity. Language models at the layer of objects and above remain vague. After one year of age, language model learning speeds up, language models become less vague and more specific much faster than the corresponding cognitive models. This is especially true about contents of abstract models, which cannot be directly perceived by the senses, such as “law,” “state,” “rationality.” This is the neural mechanism by which kids by the age of five can talk about most of contents of the surrounding culture but cannot function like adults: language models are acquired ready-made from the surrounding language, but cognitive models remain vague and gradually acquire concrete contents throughout life. This is the neural mechanism colloquially called “acquiring experience.” Language can be learned fast because language models exist “ready-made” in the surrounding language at all levels of the mind hierarchy. Cognition, consisting in mental representations of sets of objects, experiences, etc., on the opposite, requires experience with real world.

Human learning of cognitive models continues through the lifetime and is guided by language models. The knowledge instinct drives the human mind to develop more specific and concrete cognitive models by accumulating experience throughout life in correspondence with language models.

3.2 Experimental Evidence

As mentioned, experimental evidence for the dual model is almost nil. The first experimental indication has appeared in [25]. Those researchers demonstrated that categorical perception of color in prelinguistic infants is based in the right brain hemisphere. As language is acquired and access to lexical color codes becomes more automatic, categorical perception of color moves to the left hemisphere (between two and five years) and adult’s categorical perception of color is based in the left hemisphere (where language mechanisms are located).

These experiments have provided evidence for neural connections between perception and language, a foundation of the dual model. Possibly it confirms another aspect of the dual model: the crisp and conscious language part of the model hides from our consciousness vaguer cognitive part of the model. This is similar to what we observed in the closed-opened eye experiment: with opened eyes we are not conscious about vague imaginations-priming signals.

So, we can answer some of the questions posed at the beginning of the section. Language and cognition are separate and closely related mechanisms of the mind. They evolve jointly in ontological development and learning, and possibly these abilities evolved jointly in evolution—this we address in more details in the next section. This joint evolution of dual models from vague to more crisp content resolves the puzzle of associationism: there is no need to learn correct associations among combinatorially large number of possible associations, words and objects are associated all the time while their concrete contents emerge in the mind.

3.3 Dual Hierarchy

In the mind hierarchy contents of lower layers are perceptual elements, objects, higher up are relationships among objects, situations, more and more abstract and general model-concepts..., and near the top are the most general concepts of the purpose and meaning of life [56], [58], [62],[63], [76], [41]. The dual model implies two parallel heterarchies of language and cognition, as illustrated in Fig. 2. Deacon (1997) suggested that the hierarchy sets the human mind apart from the animal world. The mathematical reasons why the hierarchy can only exist as a joint dual hierarchy of language and cognition is as follows. Only at the lower layers in the hierarchy cognitive models can be learned by direct perception of the world. Learning is grounded.

Learning is grounded in “real” objects. At higher levels, however, learning of cognitive models has no ground. In artificial intelligence it was long recognized that learning without grounding could easily go wrong, learned or invented models may correspond to nothing real or useful [44]. Thus, the mechanism of the dual model sets the human mind apart from the rest of animal world.

To connect the two hierarchies, the KI has to be differentiated, each word has to be emotionally connected to the corresponding concept. Unemotional language creates no motivation to develop the corresponding cognitive concept.

3.4 Emotionality of Language and Meanings

What is a source of emotionality in languages? Language and voice started separating from ancient emotional centers possibly millions of years ago. Nevertheless, emotions are present in language. Most of these emotions originate in cortex and are controllable aesthetic emotions. Their role in satisfying the knowledge instinct is considered in the next section. Emotional centers in cortex are neurally connected to old emotional limbic centers, so both influences are present. Emotionality of languages is carried in language sounds, what linguists call prosody or melody of speech. This ability of human voice to affect us emotionally is most pronounced in songs. Songs and music, however, is a separate topic, not addressed in this paper.

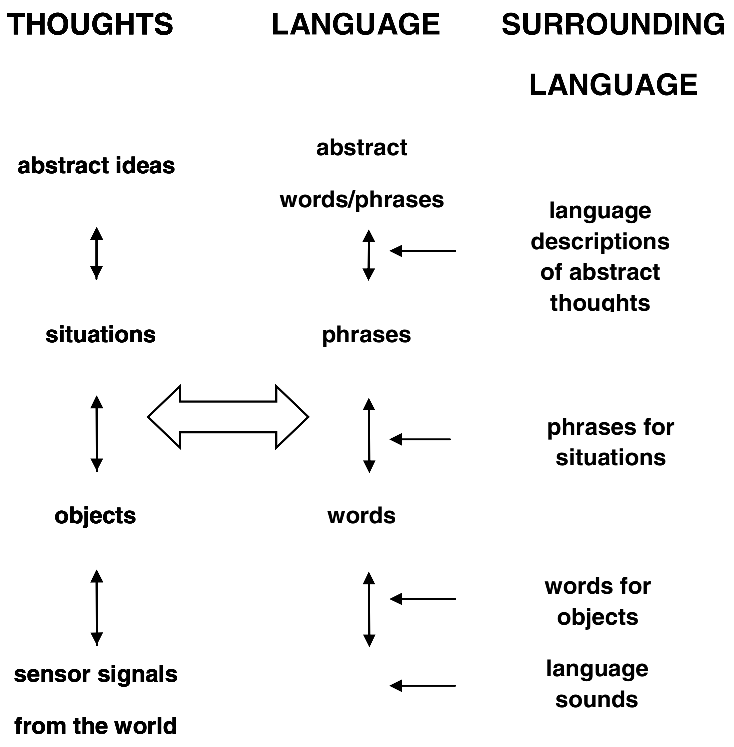


Fig. 2. Hierarchical integrated language-cognition NMF system. "Heterarchy" refers to cross-layer connections, not shown, and to the consequence that the hierarchical structure is not logically strict as may appear from the figure. At each layer in a hierarchy there are integrated language and cognition models (thick arrow). Similarities are integrated as products of language and cognition similarities. Initial models are fuzzy placeholders, so integration of language and cognition is sub-conscious. Association variables depend on both language and cognitive models and signals. Therefore language model learning helps learning cognitive models.

"The right level" of emotionality is crucial for developing cognitive parts of models. If language parts of models are highly emotional, any deliberate discourse is impossible and there will be no room for language and cognitive development (as among primates). If language parts of models are non-emotional at all, there would be no motivational force to engage into conversations, to develop language models, and motivation for developing higher cognitive models will be reduced. Lower cognitive models, say for object perception will be developed, first, because they are imperative for survival, and second, because they can be developed independently from language, based on direct sensory perceptions, like in animals. But models of situations and higher cognition are developed based on language models [60], [62], [64], [66], [72]. As discussed later, this requires emotional connections between cognitive and language models.

Primordial fused language-cognition-emotional models have differentiated long ago, involuntary connections between voice-emotion-cognition dissolved with emergence of language. They were replaced with habitual connections. Sounds of all languages have changed, nevertheless, if sounds of a language changes slowly, connections between sounds and meanings persists, it follows that emotion-meaning connections persist. This persistence is a foundation of meanings, because meanings imply motivations. If sounds of language changes too fast, cognitive models are severed from motivations, and meanings disappear. If sound changes too slowly, it nails meanings emotionally to old ways, and culture stagnates.

Therefore the next step toward understanding cultural evolution is to identify mechanisms determining changes of language sound. Changes in language sounds are controlled by grammar. In inflectional languages, affixes and endings are fused with sounds of word roots. Pronunciation-sounds of affixes are controlled by few rules, which persist over thousands of words. These few rules are manifest in every phrase. Therefore every child learns to pronounce them correctly. Positions of vocal tract and mouth muscles for pronunciation of affixes are fixed throughout population and are conserved throughout generations. Correspondingly, pronunciation of whole words cannot vary too much, and language sound changes slowly. Inflections therefore play a role of "tail that wags the dog," they anchor language sounds and preserve meanings. This, I think, Humboldt (1836/1967) meant by "firmness" of inflectional languages. When inflections disappear, this anchor is no more, nothing prevents sound of language to become fluid and change with every generation.

This has happened with English language after transition from Middle English to Modern English [40], most of inflections have disappeared and sound of the language started changing within each generation, this process continues today. English evolved into a powerful tool of cognition unencumbered by excessive emotionality, English language spread democracy and technology around the world. This was made possible by conceptual differentiation empowered by language, which overtook emotional synthesis. But the loss of synthesis has also lead to ambiguity of meanings and values. Current English language cultures face internal crises, uncertainty about meanings and purposes. Many people cannot coupe with diversity of life. Future research in psycholinguistics, anthropology, history, historical and comparative linguistics, and cultural studies will examine interactions between languages and cultures. Experimental evidence suggests emotional differences among languages consistent with our hypothesis [31], [32].

Neural mechanisms of grammar, language sound, related emotions-motivations, and meanings hold a key to connecting neural mechanisms in the individual brains to evolution of cultures. Studying them experimentally is a challenge for future research. It is not even so much a challenge, because experimental methodologies are at hand, they just should be applied to these issues. Following sections develop mathematical models based on existing evidence that can guide this future research.

4 Cultural Dynamics

Mathematical models of the mind mechanisms corresponding to the discussions in previous sections determine evolution of cultures. These models are based on the available experimental evidence and theoretical development by many authors summarized in [47], [49], [52], [54], [57], [59], [62], [63], [64], [65], [66], [71], [73], [88] and it corresponds to experimental data [31], [2].

The hierarchical dynamics of KI manifests as differentiation and synthesis. Differentiation drives creation of concrete, specific concepts (top-down evolution of the hierarchy). Synthesis drives creation of general concept-models, unifying differentiated signals (bottom-up evolution).

They are in complex relationships, at once symbiotic and antagonistic [65], [73], [75]. Synthesis creates emotional value of knowledge, it unifies language and cognition, creates conditions for differentiation, it leads to spiritual inspiration, to active creative behavior leading to fast differentiation, to creation of knowledge, to science and technology. At the same time, a "too high" level of synthesis, high emotional values of concepts stifles differentiation, as in traditional consciousness.

Synthesis, *S*, might lead to growth of general concept-models, and to growth of the hierarchy. This is counterbalanced by differentiation, the number of concepts grows, leading to "precise knowledge about nothing." In the knowledge-acquiring regime, growth of synthesis is limited psychologically, emotions of KI satisfaction, when "spread" over large number of concepts would not sustain growth in the concept number, *D*. This is well known in many engineering problems, when too many models are used. Thus, whereas emotional synthesis creates a condition for differentiation (high emotional value of knowledge, efficient dual model connecting language and cognition), conceptual differentiation undermines synthesis (value of knowledge, *S,* and its diversity, *D*, fall). This interaction can be modeled by the following equations:

$$dD/dt=aDG(S),\ G(S)=(S - S_0)\ exp(-(S-S_0)/ S_1),$$
$$dS/dt = -b\ D + d\ H,$$
$$H(t) = H_0 + e^*t.$$

Here, t is time, D is a number of concepts (differentiation), S models synthesis, emotional satisfaction of the knowledge instinct, H is a number of hierarchical levels, a, b, d, e, S_0 and S_1 are constants. Differentiation, D, grows proportionally to already existing number of concepts, as long as this growth is supported by synthesis, while synthesis is maintained at a "moderate" level, $S_0 < S < S_1$. "Too high" level of synthesis, $S > S_1$, stifles differentiation by creating too high emotional value of concepts. Synthesis, S, grows in the hierarchy, along with a number of hierarchical levels, H. By creating emotional values of knowledge, it sustains differentiation, however, differentiation, by spreading emotions among a large number of concepts destroys synthesis. Hierarchical dynamics H we consider over a period of slow growth of the hierarchy H. At moderate values of synthesis, solving eqs.(2) yields a solution in Fig. 3. The number of concepts grows until certain level, when it results in reduction of synthesis, then the number of models falls. As a number of models falls, synthesis grows, and the growth in models resumes. The process continues with slowly growing, oscillating number of models (knowledge-accumulating consciousness).

Another solution corresponds to initially high level of synthesis, Fig. 4. Synthesis continues growing whereas differentiation levels off. This leads to a more and more stable society with high synthesis, in which high emotional values are attached to every concept, however, differentiation stagnates.

These two solutions can be compared to Humboldt's (1836/1967) characterization of languages and cultures. He contrasted inert objectified "outer form" of words vs.

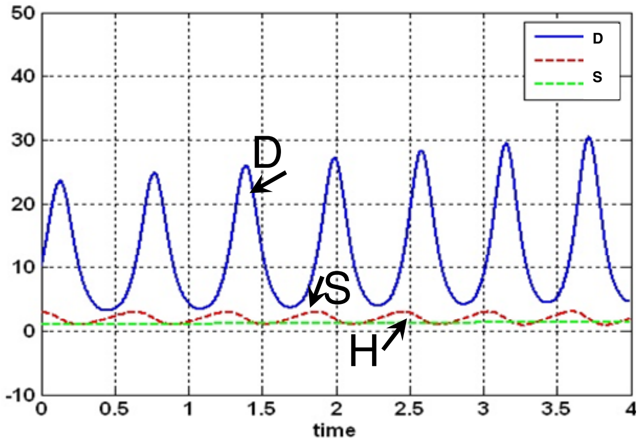


Fig. 3. Evolution of culture at moderate values of synthesis oscillates: periods of flourishing and knowledge accumulation alternate with collapse and loss of knowledge (*a = 10, b = 1, d = 10, e = 0.1,* S_0*=2,* S_1*=10*, and initial values *D(t=0) = 10, S(t=0) = 3,* H_0 *= 1*, parameter and time units are arbitrary). In long time the number of models slowly accumulates, this corresponds to slowly growing hierarchy.

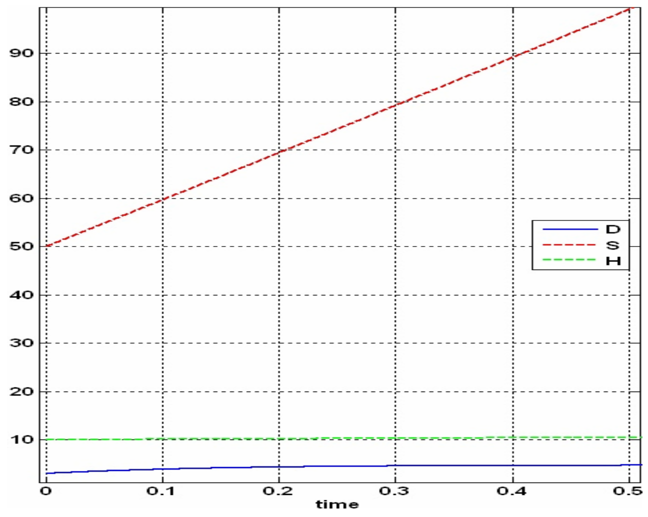


Fig. 4. Evolution of highly stable, stagnating society with growing synthesis. High emotional values are attached to every concept, while knowledge accumulation stops (*D(t=0)= 3,* H_0 *= 10, S(t=0) = 50,* S_0 *= 1,* S_1 *= 10, a = 10, b = 1, d = 10, e=1*).

subjective, culturally conditioned, and creative "inner form." Humboldt's suggestion continues to stir linguists' interest today, yet seem mysterious and not understood scientifically.

The above analysis suggests the following interpretation of Humboldt's intuitions in terms of neural mechanisms. His "inner form" corresponds to the integrated, moderately emotional neural dual model [60], [64], [66], [72], [73]. Contents of cognitive models are being developed guided by language models, which accumulate cultural wisdom. "Outer form" of language corresponds to inefficient state of neural dual model, in which language models do not guide differentiation of the cognitive ones. This might be due to either too strong or too weak involvement of emotions. If emotional involvement in cognition or language is too weak, learning does not take place because motivation disappears. If emotional involvement is too strong, learning does not take place because old knowledge is perceived as too valuable, and no change is possible. The first case might be characteristic of low-inflected languages, when sound of language changes "too fast," and emotional links between sound and meanings are severed. The second case might be characteristic of "too strongly" inflected languages, in which sound changes "too slowly" and emotions are connected to meanings "too strongly," this could be a case of Fig. 4. A brief look at cultures and languages certainly points to many examples of this case: highly inflected languages and correspondingly "traditional" stagnating cultures.

Much of contemporary world is "too flat" for a single language and culture, existing without outside influences. When two cultures interact and exchange differentiation and synthesis, eventually both cultures stabilize each other, both benefited from fast growth and reduced instabilities [67], [73].

5 Future Research

The dual model implies a relatively minimal neural change from the animal to the human mind, which have been prepared by the mechanism of mirror neurons. DL resolves a long-standing mystery of how human language, thinking, and culture could have evolved in a seemingly single big step, too large for an evolutionary mutation, too fast and involving too many advances in language, thinking, and culture, happening almost momentarily around 50,000 years ago [14], [45]. DL along with the dual model explains how changes, which seem to involve improbable steps according to logical intuition, actually occur through continuous dynamics. The proposed theory provides a mathematical basis for concurrent emergence of hierarchical human language and cognition. The dual model is supported by experimental evidence [31], [25], and future experiments can test it directly. Next theoretical steps would further develop this theory in multi-agent simulations, leading to more specific and experimentally testable predictions in language and cultural evolution.

Classic ideas of frames and schemas were demonstrated to face combinatorial complexity, and this is the fundamental reason for their failures despite decades of development. The proposed theory produces conceptual relations, binding, and recursion through the mechanisms of the hierarchy, in addition it explains how language acquires meanings in connecting with cognition due to the mechanism of the dual model. These predictions are experimentally testable and several groups of psychologists and neuroscientists are working in these directions.

This chapter also *challenges* the idea of arbitrariness of vocalization. It is suggested that a significant degree of arbitrariness in current languages is a distal result of millennia of language evolution in presence of the dual model. Instead of assuming arbitrariness as fundamental, future research should concentrate on its emergence from primordial fusion of sound and emotion. We assume that origins of language are contemporaneous with origins of music, primordial undifferentiated vocalizations split in two: first, more semantic and less emotional mechanisms of language, and second, less semantic and more emotional mechanisms of music [70]. Whereas language differentiates consciousness, music unifies conscious and unconscious into a whole self. It is my hope that a wide field of experimental research directions opened by this discussion will be explored (several experimental programs have already started).

Connections between the neural mechanisms, language, emotions, and cultural evolution proposed in this paper are but a first step requiring much experimental evidence and theoretical development. Influence of languages on cultures, the Bhartrihari-Humboldt-Nietzsche-Sapir-Whorf hypothesis [4], [34], [46], [96], [102] formalized by the discussed mechanisms adds a novel aspect to this old idea, emotional contents of languages could be more important in influence on cultures than their conceptual contents.

Neural mechanisms proposed in this paper and models inspired by these mechanisms, although simple, nevertheless result in concrete predictions relating language grammars and types of cultures. These predictions can be verified in psycholinguistic laboratories, eq. (2) coefficients can be measured using existing methods.

Future mathematical-theoretical research should address development of multi-agent simulations, connecting neural and cultural mechanisms of emotions and cognition and their evolution mediated by language. The KI theory should be developed toward theoretical understanding of its differentiated forms explaining multiplicity of aesthetic emotions in language prosody and music [61], [64], [71], [73], [75]. This theoretical development should go along with experimental research clarifying neural mechanisms of KI [41] and the dual language-cognitive model, [72].

Recent experimental results on neural interaction between language and cognition [25], [98] support the mechanism of the dual model, they should be expanded to interaction of language with emotional-motivational, voicing, behavioral, and cognitive systems.

Prehistoric anthropology should evaluate the proposed hypothesis that the primordial fused system of conceptual cognition, emotional evaluation, voicing, motivation, and behavior differentiated at different prehistoric time periods—are there data to support this hypothesis, can various stages of prehistoric cultures be associated with various neural differentiation stages? Can different humanoid lineages be associated with different stages of neural system differentiation. What stage of neural differentiation corresponds to Mithen's hypothesis about singing Neanderthals [45]? Psychological social and anthropologic research should go in parallel documenting various cultural evolutionary paths and correlations between cognitive and emotional contents of historical and contemporary cultures and languages.

Proposed correlation between grammar and emotionality of languages can be verified in direct experimental measurements using skin conductance and fMRI neuro-imaging. Emotional version of Sapir-Whorf hypothesis should be evaluated in parallel psychological and anthropological research. More research is needed to document cultures stagnating due to "too" emotional languages, as well as crises of lost values due to "low" emotionality of language in English-speaking countries.

Acknowledgements. I am thankful to M. Alexander, M. Bar, R. Brockett, M. Cabanac, R. Deming, F. Fontanari, R. Ilin, F. Lin, J. Gleason, R. Kozma, D. Levine, A. Ovsich, and B. Weijers, and to AFOSR PMs Drs. J. Sjogren, D. Cochran, and J. Zhang for partial support of this research.

References

1. Arbib, M.A.: From monkey-like action recognition to human language: An evolutionary framework for neurolinguistics. Behavioral and Brain Sciences 28, 105–167 (2005)
2. Bar, M., Kassam, K.S., Ghuman, A.S., Boshyan, J., Schmid, A.M., Dale, A.M., Hämäläinen, M.S., Marinkovic, K., Schacter, D.L., Rosen, B.R., Halgren, E.: Top-down facilitation of visual recognition. Proceedings of the National Academy of Sciences USA, 103, 449–454 (2006)
3. Barsalou, L.W.: Perceptual symbol systems. Behavioral and Brain Sciences 22, 577–660 (1999)
4. Bhartrihari (IVCE/1971). The Vâkyapadîya, Critical texts of Cantos I and II with English Translation. Trans. K. Pillai. Delhi: Motilal Banarsidass
5. Brighton, H., Smith, K., Kirby, S.: Language as an evolutionary system. Phys. Life Rev. 2(3), 177–226 (2005)
6. Cabanac, M.: What is emotion? Behav. Proc., 60, 69–84 (2002), doi:10.1016/S0376-6357(02)00078-5
7. Cacioppo, J.T., Petty, R.E.: The need for cognition. Journal of Personality and Social Psychology 42, 116–131 (1982)
8. Carpenter, G.A., Grossberg, S.: A massively parallel architecture for a self-organizing neural pattern recognition machine. Computer Vision, Graphics, and Image Processing 37, 54–115 (1987)
9. Chomsky, N.: Aspects of the theory of syntax. MIT Press, Cambridge (1965)
10. Chomsky, N.: The minimalist program. MIT Press, Cambridge (1995)
11. Christiansen, M.H., Chater, N.: Language as shaped by the brain. Behavioral and Brain Sciences 31(5), 489–509 (2008)
12. Christiansen, M.H., Kirby, S.: Language evolution. Oxford Univ. Press, New York (2003)
13. Croft, W., Cruse, D.A.: Cognitive linguistics. Cambridge University Press, Cambridge (2004)
14. Deacon, T.W.: The symbolic species: the co-evolution of language and the brain. Norton, New York (1997)
15. Deming, R., Perlovsky, L.I.: A Mathematical Theory for Learning, and its Application to Time-varying Computed Tomography. New Math. and Natural Computation 1(1), 147–171 (2005)

16. Deming, R.W., Perlovsky, L.I.: Concurrent multi-target localization, data association, and navigation for a swarm of flying sensors. Information Fusion 8, 316–330 (2007)
17. Deming, R., Schindler, J., Perlovsky, L.: Multitarget/Multisensor Tracking using only Range and Doppler Measurements. IEEE Transactions on Aerospace and Electronic Systems 45(2), 593–611 (2009)
18. Evans, V., Green, M.: Cognitive linguistics: an introduction. Edinburgh University Press, Edinburgh (2006)
19. Fauconnier, G., Turner, M.: The origin of language as a product of the evolution of modern cognition. In: Laks, B., et al. (eds.) Origin and Evolution of Languages: Approaches, Models, Paradigms, Equinox, London (2008)
20. Fontanari, J.F., Perlovsky, L.I.: Solvable null model for the distribution of word frequencies. Physical Review E 70, 042901 (2004)
21. Fontanari, J.F., Perlovsky, L.I.: Evolving Compositionality in Evolutionary Language Games. IEEE Transactions on Evolutionary Computations 11(6), 758–769 (2007), doi:10.1109/TEVC.2007.892763
22. Fontanari, J.F., Perlovsky, L.I.: A game theoretical approach to the evolution of structured communication codes. Theory in Biosciences 127(3), 205–214 (2008a)
23. Fontanari, J.F., Perlovsky, L.I.: How language can help discrimination in the Neural Modeling Fields framework. Neural Networks 21(2-3), 250–256 (2008b)
24. Fontanari, F.J., Tikhanoff, V., Cangelosi, A., Ilin, R., Perlovsky, L.I.: Cross-situational learning of object–word mapping using Neural Modeling Fields. Neural Networks 22(5-6), 579–585 (2009)
25. Franklin, A., Drivonikou, G.V., Bevis, L., Davie, I.R.L., Kay, P., Regier, T.: Categorical perception of color is lateralized to the right hemisphere in infants, but to the left hemisphere in adults. PNAS 105(9), 3221–3225 (2008)
26. Gnadt, W., Grossberg, S.: SOVEREIGN: An autonomous neural system for incrementally learning planned action sequences to navigate towards a rewarded goal. Neural Networks 21, 699–758 (2008)
27. Gödel, K.: Collected works. In: Feferman, S., Dawson Jr., J.W., Kleene, S.C. (eds.) Publications 1929-1936, vol. I. Oxford Univ. Press, New York (1931/1994)
28. Grossberg, S.: Neural Networks and Natural Intelligence. MIT Press, Cambridge (1988)
29. Grossberg, S., Levine, D.S.: Neural dynamics of attentionally modulated Pavlovian conditioning: blocking, inter-stimulus interval, and secondary reinforcement. Psychobiology 15(3), 195–240 (1987)
30. Grossberg, S., Seidman, D.: Neural dynamics of autistic behaviors: Cognitive, emotional, and timing substrates. Psychological Review 113, 483–525 (2006)
31. Guttfreund, D.G.: Effects of language usage on the emotional experience of Spanish-English and English-Spanish bilinguals. J. Consult Clin. Psychol. 58, 604–607 (1990)
32. Harris, C.L., Ayçiçegi, A., Gleason, J.B.: Taboo words and reprimands elicit greater autonomic reactivity in a first language than in a second language. Applied Psycholinguistics 24, 561–579 (2003)
33. Hauser, M.D., Chomsky, N., Fitch, W.T.: The faculty of language: what is it, who has it, and how did it evolve? Science 298, 1569–1579 (2002)
34. von Humboldt, W.: Über die Verschiedenheit des menschlichen Sprachbaues und ihren Einfluss auf die geistige Entwickelung des Menschengeschlechts. In: Dummler, F., Lehmann, W.P. (eds.) A Reader in Nineteenth Century Historical Indo-European Linguistics. Indiana University Press, Bloomington (1967)

35. Hurford, J.: The evolution of human communication and language. In: D'Ettorre, P., Hughes, D. (eds.) Sociobiology of communication: an interdisciplinary perspective, pp. 249–264. Oxford University Press, New York (2008)
36. Ilin, R., Perlovsky, L.I.: Cognitively Inspired Neural Network for Recognition of Situations. International Journal of Natural Computing Research 1(1), 36–55 (2010)
37. Juslin, P.N., Västfjäll, D.: Emotional responses to music: The Need to consider underlying mechanisms. Behavioral and Brain Sciences 31, 559–575 (2008)
38. Kay, P.: An informal sketch of a formal architecture for construction grammar. Grammars 5, 1–19 (2002)
39. Kozma, R., Puljic, M., Perlovsky, L.: Modeling goal-oriented decision making through cognitive phase transitions. New Mathematics and Natural Computation 5(1), 143–157 (2009)
40. Lerer, S.: Inventing English. Columbia University Press, Chichester (2007)
41. Levine, D.S., Perlovsky, L.I.: Neuroscientific insights on Biblical myths: Simplifying heuristics versus careful thinking: Scientific analysis of millennial spiritual issues. Zygon, Journal of Science and Religion 43(4), 797–821 (2008)
42. Linnehan, R., Mutz, C., Perlovsky, L.I., Weijers, B., Schindler, J., Brockett, R.: Detection of patterns below clutter in images. In: Int. Conf. Integration of Knowledge Intensive Multi-Agent Systems, Cambridge, MA (2003)
43. Mayorga, R., Perlovsky, L.I. (eds.): Sapient Systems. Springer, London (2008)
44. Meystel, A.M., Albus, J.S.: Intelligent systems: Architecture, Design, and Control. Wiley, New York (2001)
45. Mithen, S.: A creative explosion? Theory of mind, language, and the disembodied mind of the Upper Paleolithic. In: Mithen, Steven (eds.) Creativity in Human Evolution and Prehistory, pp. 165–191. Routledge, London (1998)
46. Nietzsche, F.: Untimely Meditations. Tr. Hollingdale. Cambridge Univ. Press, Cambridge (1876/1983)
47. Perlovsky, L.I.: Multiple Sensor Fusion and Neural Networks. In: DARPA Neural Network Study. MIT/Lincoln Laboratory, Lexington (1987)
48. Perlovsky, L.I.: Cramer-Rao Bounds for the Estimation of Normal Mixtures. Pattern Recognition Letters 10, 141–148 (1989)
49. Perlovsky, L.I.: Computational Concepts in Classification: Neural Networks, Statistical Pattern Recognition, and Model Based Vision. Journal of Mathematical Imaging and Vision 4(1), 81–110 (1994a)
50. Perlovsky, L.I.: A Model Based Neural Network for Transient Signal Processing. Neural Networks 7(3), 565–572 (1994b)
51. Perlovsky, L.I.: Cramer-Rao Bound for Tracking in Clutter and Tracking Multiple Objects. Pattern Recognition Letters 18(3), 283–288 (1997)
52. Perlovsky, L.I.: Physical Concepts of Intellect. Proceedings of Russian Academy of Sciences 354(3), 320–323 (1997)
53. Perlovsky, L.I.: Cramer-Rao Bound for Tracking in Clutter and Tracking Multiple Objects. Pattern Recognition Letters 18(3), 283–288 (1997)
54. Perlovsky, L.I.: Conundrum of Combinatorial Complexity. IEEE Trans. PAMI 20(6), 666–670 (1998)
55. Perlovsky, L.I.: Cramer-Rao Bound for Tracking in Clutter. In: Streit, R.L. (ed.) Probabilistic Multi-Hypothesis Tracking, pp. 77–84. NUWC Press, Newport (1998)
56. Perlovsky, L.I.: Beauty and mathematical Intellect. Zvezda (9), 190–201 (2000) (Russian)
57. Perlovsky, L.I.: Neural Networks and Intellect: using model based concepts. Oxford University Press, New York (2001)

58. Perlovsky, L.I.: Aesthetics and mathematical theories of intellect. Iskusstvoznanie 2(02), 558–594 (2002) (Russian)
59. Perlovsky, L.I.: Statistical Limitations on Molecular Evolution. Journal of Biomolecular Structure & Dynamics 19(6), 1031–1043 (2002)
60. Perlovsky, L.I.: Integrating language and cognition. IEEE Connections 2(2), 8–12 (2004)
61. Perlovsky, L.I.: Evolving agents: Communication and cognition. In: Gorodetsky, V., Liu, J., Skormin, V.A. (eds.) AIS-ADM 2005. LNCS (LNAI), vol. 3505, pp. 37–49. Springer, Heidelberg (2005)
62. Perlovsky, L.I.: Toward physics of the mind: concepts, emotions, consciousness, and symbols. Physics of Life Reviews 3, 23–55 (2006a)
63. Perlovsky, L.I.: Fuzzy Dynamic Logic. New Math. and Natural Computation 2(1), 43–55 (2006b)
64. Perlovsky, L. I.: Music–the first principles. Musical Theater (2006c), `http://www.ceo.spb.ru/libretto/kon_lan/ogl.shtml`
65. Perlovsky, L.I.: Evolution of languages, consciousness, and cultures. IEEE Computational Intelligence Magazine 2(3), 25–39 (2007a)
66. Perlovsky, L.I.: Modeling Field Theory of Higher Cognitive Functions. In: Loula, A., Gudwin, R., Queiroz, J. (eds.) Artificial Cognition Systems, pp. 64–105. Idea Group, Hershey (2007b)
67. Perlovsky, L.I.: Symbols: Integrated Cognition and Language. In: Gudwin, R., Queiroz, J. (eds.) Semiotics and Intelligent Systems Development, pp. 121–151. Idea Group, Hershey (2007c)
68. Perlovsky, L.I.: Neural Networks, Fuzzy Models and Dynamic Logic. In: Köhler, R., Mehler, A. (eds.) Aspects of Automatic Text Analysis (Festschrift in Honor of Burghard Rieger), pp. 363–386. Springer, Germany (2007d)
69. Perlovsky, L.I.: Neural Dynamic Logic of Consciousness: the Knowledge Instinct. In: Perlovsky, L.I., Kozma, R. (eds.) Neurodynamics of Higher-Level Cognition and Consciousness. Springer, Heidelberg (2007e)
70. Perlovsky, L.I.: Sapience, Consciousness, and the Knowledge Instinct (Prolegomena to a Physical Theory). In: Mayorga, R., Perlovsky, L.I. (eds.) Sapient Systems. Springer, London (2008a)
71. Perlovsky, L.I.: Music and consciousness, Leonardo. Journal of Arts, Sciences and Technology 41(4), 420–421 (2008b)
72. Perlovsky, L.I.: Language and Cognition. Neural Networks 22(3), 247–257 (2009a), doi:10.1016/j.neunet.2009.03.007
73. Perlovsky, L.I.: Language and Emotions: Emotional Sapir-Whorf Hypothesis. Neural Networks 22(5-6), 518–526 (2009b), doi:10.1016/j.neunet.2009.06.034
74. Perlovsky, L.I.: 'Vague-to-Crisp' Neural Mechanism of Perception. IEEE Trans. Neural Networks 20(8), 1363–1367 (2009c)
75. Perlovsky, L.I.: Musical emotions: Functions, origin, evolution. Physics of Life Reviews 7(1), 2–27 (2010a), doi:10.1016/j.plrev.2009.11.001
76. Perlovsky, L.I.: Intersections of Mathematical, Cognitive, and Aesthetic Theories of Mind. Psychology of Aesthetics, Creativity, and the Arts 4(1), 11–17 (2010b), doi:10.1037/a0018147
77. Perlovsky, L.I.: Neural Mechanisms of the Mind, Aristotle, Zadeh, & fMRI. IEEE Trans. Neural Networks 21(5), 718–733 (2010c)
78. Perlovsky, L.I.: The Mind is not a Kludge. Skeptic 15(3), 51–55 (2010d)
79. Perlovsky, L.I., Coons, R.P., Streit, R.L., Luginbuhl, T.E., Greineder, S.: Application of MLANS to Signal Classification. Journal of Underwater Acoustics 44(2), 783–809 (1994)

80. Perlovsky, L.I., Chernick, J.A., Schoendorf, W.H.: Multi-Sensor ATR and Identification Friend or Foe Using MLANS. Neural Networks 8(7/8), 1185–1200 (1995)
81. Perlovsky, L.I., Deming, R.W.: Neural Networks for Improved Tracking. IEEE Trans. Neural Networks 18(6), 1854–1857 (2007)
82. Perlovsky, L.I., Ilin, R.: Neurally and Mathematically Motivated Architecture for Language and Thought. Special Issue Brain and Language Architectures: Where We are Now? The Open Neuroimaging Journal 4, 70–80 (2010), `http://www.bentham.org/open/tonij/`
83. Perlovsky, L.I., Ilin, R.: Computational Foundations for Perceptual Symbol System. In: WCCI 2010, Barcelona (2010) (submitted for journal publication)
84. Perlovsky, L.I., Jaskolski, J.V.: Maximum Likelihood Adaptive Neural Controller. Neural Networks 7(4), 671–680 (1994)
85. Perlovsky, L.I., Kozma, R. (eds.): Neurodynamics of Higher-Level Cognition and Consciousness. Springer, Heidelberg (2007a)
86. Perlovsky, L., Kozma, R.: Editorial - Neurodynamics of Cognition and Consciousness. In: Perlovsky, L., Kozma, R. (eds.) Neurodynamics of Cognition and Consciousness. Springer, Heidelberg (2007b)
87. Perlovsky, L.I., Marzetta, T.L.: Estimating a Covariance Matrix from Incomplete Independent Realizations of a Random Vector. IEEE Trans. on SP 40(8), 2097–2100 (1992)
88. Perlovsky, L.I., Mayorga, R.: Preface. In: Mayorga, R., Perlovsky, L.I. (eds.) Sapient Systems. Springer, London (2008)
89. Perlovsky, L.I., McManus, M.M.: Maximum Likelihood Neural Networks for Sensor Fusion and Adaptive Classification. Neural Networks 4(1), 89–102 (1991)
90. Perlovsky, L.I., Plum, C.P., Franchi, P.R., Tichovolsky, E.J., Choi, D.S., Weijers, B.: Einsteinian Neural Network for Spectrum Estimation. Neural Networks 10(9), 1541–1546 (1997a)
91. Perlovsky, L.I., Schoendorf, W.H., Burdick, B.J., Tye, D.M.: Model-Based Neural Network for Target Detection in SAR Images. IEEE Trans. on Image Processing 6(1), 203–216 (1997b)
92. Perlovsky, L.I., Schoendorf, W.H., Garvin, L.C., Chang, W.: Development of Concurrent Classification and Tracking for Active Sonar. Journal of Underwater Acoustics 47(2), 375–388 (1997c)
93. Perlovsky, L.I., Schoendorf, W.H., Tye, D.M., Chang, W.: Concurrent Classification and Tracking Using Maximum Likelihood Adaptive Neural System. Journal of Underwater Acoustics 45(2), 399–414 (1995)
94. Perlovsky, L.I., Webb, V.H., Bradley, S.R., Hansen, C.A.: Improved ROTHR Detection and Tracking Using MLANS. AGU Radio Science 33(4), 1034–1044 (1998)
95. Pinker, S.: The language instinct: How the mind creates language. William Morrow, New York (1994)
96. Sapir, E.: Culture, Language and Personality: Selected Essays by Edward Sapir. University of California Press, Berkeley (1985)
97. Sebe, I.O., You, S., Neumann, U.: Globally optimum multiple object tracking. In: Masten, M.K., Stockum, L.A. (eds.) Proceedings of SPIE, Acquisition, Tracking, and Pointing XIX, vol. 5810, pp. 82–93 (2005)
98. Simmons, W.K., Stephan, B.H., Carla, L.H., Xiaoping, P.H., Barsalou, L.W.: fMRI evidence for word association and situated simulation in conceptual processing. Journal of Physiology - Paris 102, 106–119 (2008)

99. Singer, R.A., Sea, R.G., Housewright, R.B.: Derivation and evaluation of improved tracking filters for use in dense multitarget environments. IEEE Transactions on Information Theory 20, 423–432 (1974)
100. Tikhanoff, V., Fontanari, J.F., Cangelosi, A., Perlovsky, L.I.: Language and cognition integration through modeling field theory: Category formation for symbol grounding. In: Kollias, S.D., Stafylopatis, A., Duch, W., Oja, E. (eds.) ICANN 2006. LNCS, vol. 4131, pp. 376–385. Springer, Heidelberg (2006)
101. Ungerer, F., Schmid, H.-J.: An introduction to cognitive linguistics. Pearson, New York (2006)
102. Whorf, B.L.: Language, Thought, and Reality. MIT Press, Cambridge (1956)

A Framework for Intelligent Analysis of Intelligence Data*

Qiang Shen and Changjing Shang

Dept. of Computer Science, Aberystwyth University, Wales, U.K.
{qqs,cns}@aber.ac.uk

Abstract. Intelligent data analysis plays an important role in many application domains. In particular, for detection and prevention of serious crime, including terrorist activity, automated modelling and analysis of intelligence data is of great practical significance. Such an intelligent analysis system will provide useful decision support for intelligence analysts, offering an effective means in the assessment of possible scenarios given observations which may be imprecise and/or uncertain. Intelligent analysis of intelligence data will therefore help to facilitate rapid response in devising and deploying preventive measures. This paper presents an initial approach to the development of a general framework that integrates key component systems for intelligent data analysis, with an application focus on intelligence data. It describes the functionalities of the important component systems and introduces example techniques that are useful to implement such systems. The paper also discusses major challenges and opportunities for further relevant research.

1 Introduction

Effective automated analysis of data has found successful real-world applications in many areas, ranging from science discovery [28] and knowledge enrichment [5], through plant monitoring [48] and crime investigation [1], to medical diagnosis [43] and systems biology [33]. Many approaches to intelligent data analysis have been developed. One typical aim of these approaches is to assist in solving complex real-world problems which involves timely and intelligent decision-making processes, through analysis of a large volume of available information. This is particularly the case in the domain of intelligence data analysis. For example, in the wake of terrorist atrocities such as September 11, 2001, and July 7, 2005, intelligence analysts have commented that the failure in the detection of terrorist activity is not necessarily due to lack of data, but to difficulty in relating and interpreting the available intelligence on time [49].

In analysis of intelligence data, it is often observed that most criminal and terrorist organisations form flexible networks of loosely related individuals and sub-organisations. These networks are often embedded within legitimate society and remain secrete. Yet,

* This work was partly supported by UK EPSRC grants GR/S63267/01-02, GR/S98603/01 and EP/D057086/1, and partly by a UK Royal Academy of Engineering/Daphne Jackson Research Fellowship. The authors are grateful to all members of the project teams for their contributions, but will take full responsibility for the views expressed here.

organised crime and terrorist activity does leave a trail of information, such as captured communications and forensic evidence, which can be collected by police and security agents. Based on partial information gathered, which may be imprecise and/or uncertain, experienced intelligence analysts can suggest plausible scenarios that may explain the observations. This, in turn, may help to identify potential criminals who may pose a threat. However, the amount of intelligence data possibly relevant may well be overwhelming for human examination, especially with limited time and resources. Automated modelling and analysis through hypothetical (re-)construction of the activities that may have led to the intelligence data obtained, therefore, presents an important and challenging research topic for crime prevention and detection.

Many existing intelligent systems have been developed to enhance efforts for crime reduction. However, their effectiveness is often crucially dependent upon the experience of the intelligence analysts since any potential threat is identified ultimately only by the analysts. This vulnerability can be addressed by a data analysis tool which automatically generates plausible scenarios when given a limited amount of real or hypothesised information, and which offers the user with the means to assess the properties of such scenarios. This paper, based on the original proposals presented in [46], introduces a general framework for the development of such systems. This is in order to assist (but not to replace) intelligence analysts in identifying plausible scenarios of criminal or terrorist activity in presence of obtained intelligence data, and in assessing the reliability, risk and urgency of generated hypotheses. In particular, the paper describes the functionalities of the key component systems that may be jointly utilised to construct an integrated intelligent analysis tool. It also introduces example techniques that are useful to implement such component systems.

The rest of this paper is organised as follows. The next section presents the framework and outlines the essential components of such a data analysis system. Then, Section 3 shows particular instantiations of the techniques useful to implement the key components of this framework. Essential ideas are illustrated with some simple examples. Section 4 summarises the paper and points out important further research.

2 Intelligent Data Analysis Framework

Conventional approaches to building intelligent tools, such as rule-based and case-based systems, have found successful applications in the area of intelligence data analysis. However, their scope is restricted to either the situations foreseen or those resulting from previously encountered cases. Yet, serious crime like organised terrorist activity tends to be unique. Thus, the nature of intelligence data in such problem domains requires the intelligent analysis system to be robust, capable of working with variations of data.

Having taken notice of this, the present work employs a model-based approach to intelligent data analysis [49]. That is, the knowledge base of such a system consists of generic and reusable component parts of plausible scenarios or hypotheses, called model or scenario fragments (interchangeably). Such fragments include: types of (human and material) resources required for certain classes of crime activity, ways in which such resources may be organised, and forms of evidence that may be obtained (and hence acquired from intelligence databases) given partial observed or hypothesised scenarios.

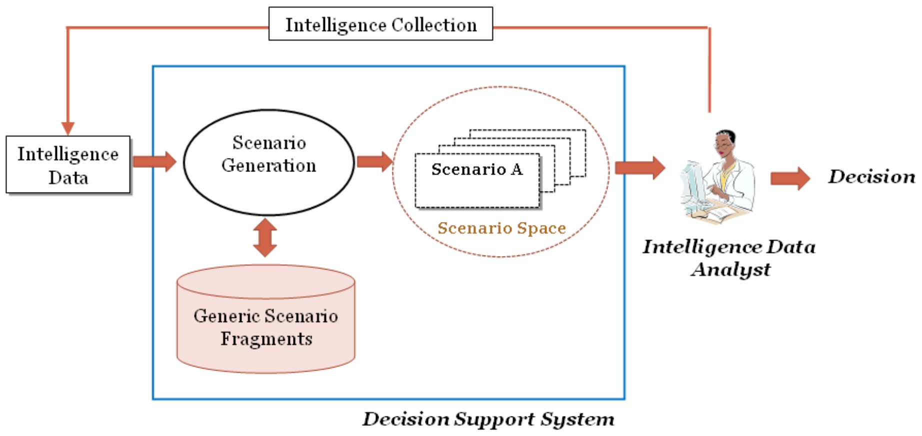


Fig. 1. Framework of Intelligent Systems for Intelligence Data Analysis

Figure 1 shows the general framework of the approach taken in this research. Based on gathered intelligence data, the scenario generation mechanism instantiates and retrieves any relevant model fragments from a library of generic scenario fragments, and combines such fragments to form plausible explanations for the data. Such a model-based approach fits well the task of coping with a wide variety of likely scenarios that may each explain the observations. A description of the key functionalities of an intelligent analysis system that follows this approach is given below.

2.1 Flexible Composition Modelling

As indicated above, the central task for automated intelligence data analysis is to establish an inference mechanism that can instantiate and then dynamically compose generic model fragments into scenario descriptions, which may explain the available data. This requires the automatic construction of computational models that are both adequate and efficient for the task. This in turn, demands the underlying modelling tool to be able to automatically construct many variations of a given type of scenario from a relatively small knowledge base. A compositional modelling approach [26] is devised for this purpose, by combining reusable model fragments on the fly, in order to ensure robustness and to deal with changing requirements.

Essentially, a compositional modeller stores hypothetical scenarios in a hypergraph or scenario space, representing states and events within different scenarios as nodes and the (causal) relations between such nodes as directed hyperarcs. A significantly extended version [51] of the assumption-based truth maintenance system (ATMS) [13] is exploited to maintain the scenario space and to enable retrieval of partial scenarios and other useful information (e.g. potential evidence to help validate a scenario) from it efficiently.

Existing compositional modelling work assumes that the model fragments within the knowledge base are expressed by precise and crisp information. It is therefore restrictive in real-world applications. In order to cope with intelligence data, consideration must be given to inexact knowledge and data captured in a variety of forms. Conceptually, inexact information may be classified into the following three general categories at least [16]:

1. *Imprecision*: It arises due to lack of sharp distinctions or boundaries between pieces of information (e.g. Bob is *tall* not *medium*). Often, a vague proposition can be modelled by a fuzzy set [61] which identifies a soft constraint on a set of elements. Instead of using a crisp partition, an element of a fuzzy set is allowed to satisfy the soft constraint to a certain degree.
2. *Uncertainty*: It depicts the reliability or confidential weight associated with a given piece of information (be it data or knowledge). Due to the involvement of uncertainty, it is difficult to tell the exact truth of a given statement. In the literature, two types of uncertainty are often referred to: randomness and fuzziness. Probability theory is typically employed to model randomness, while the possibility theory [59] is well established to deal with fuzziness (where a fuzzy set membership function is interpreted as a possibility distribution).
3. *Both vagueness and uncertainty*: That is, information of type 1 and that of type 2 coexist (e.g. The amount of collected fiber is *a_lot*, with a certainty degree of *very_likely*).

Much work has been developed to support reasoning with inexact knowledge and data (e.g. [29]). The general trend of the existing techniques is to integrate the underlying distinct pieces of inexact information into a global measure. However, in performing such integration, the underlying semantics associated with different information components may be destroyed. It is of great interest and potentially beneficial to establish a new mechanism which will maintain the associated semantics when reasoning with inexact knowledge and data. This work follows the recent development in fuzzy compositional modelling [16] in an attempt to instantiate and compose model fragments into consistent scenario descriptions when given a mixture of the aforementioned types of information.

The compositional modelling approach developed in this research differs from those in the literature in two distinct ways:

1. *Suitability* for intelligence data analysis. Fuzzy compositional modelling provides a more flexible knowledge representation formalism. Approximate and vague information may be abstractly specified in the knowledge base, e.g. a chemical being “highly explosive”. Due to the involvement of such information, a precise and certain match between the available observations and the model fragments cannot be expected in general. In these cases, the boolean retrieval approach used in existing compositional modelling work usually fails to return any fragments since they only partially match the available information. A fuzzy mechanism is able to retrieve fragments which involve no exact match but which are likely to be relevant to the collected data. This allows the generation of scenarios from a wide range of data sources, including factual data, collected intelligence, and hypothesised but unsubstantiated information. This entails matching specific data (e.g. the names of

discovered chemicals) with broader (and possibly subjective) knowledge and other imprecise information contents.

2. *Ability* to speculate about plausible relations between different cases. Often, intelligence data refers to individuals and objects whose identity is only partially specified. For example, when a person is observed on a CCTV camera, some identifying information can be collected, but this may be insufficient for an exact identification. When a person with similar features has been identified elsewhere, it is important that any relation between both sightings is explored. Ideas originally developed in the area of link-based similarity analysis [9,32] are adapted [7] for: (a) identifying similar individuals and objects in a space of plausible scenarios, and (b) supporting the generation of hypothetically combined scenarios to explore the implications of possible matches.

2.2 Intelligence Data Modelling and Analysis

Intelligent analysis of intelligence data in general and that of potential serious criminal activity (especially terrorist activity) in particular, is a non-trivial task. It is not known in advance what aspects of such activity may be observed, and how they will be interconnected. Thus, conventional approaches to intelligent data analysis, which aim to identify pre-specified patterns of data, are difficult to adapt to this domain.

Although general and potentially suitable, the model-based approach adopted here may lead to systems that generate a large number of plausible scenarios for a given problem. It is therefore necessary for such a system to incorporate a means to sort the plausible scenarios, so that the generated information remains manageable within a certain time frame. For this purpose, generated scenarios are presented to human analysts with measurements of their reliability, risk, and urgency. The reliability of a scenario estimates the likelihood of its occurrence. The risk posed by a scenario reflects the number of potential casualties and/or the degree of possible economic cost of failure to prevent such activity. A scenario's urgency corresponds to the time in which the suspect terrorist activity may be carried out.

Each of the aforementioned ranking features may be assessed by a numeric metric. However, intelligence data and hypotheses are normally too vague to produce precise estimates that are also accurate. Therefore, this work devises a novel fuzzy mechanism to provide an appropriate method of assessing and presenting the reliability, risk and urgency of generated scenarios [50]. The framework also covers additional tools such as a facility to propose additional information sources (by exploring additional, real or hypothesised, evidence that may be generated in a given scenario).

Figure 2 shows an implementation specification of the general framework given in Fig. 1. Technical modules include the following, with each associated with an introductory reference:

– Fuzzy Feature Selection carries out semantics-preserving dimensionality reduction (over nominal and real-valued data) [23].
– Fuzzy Learning provides a knowledge modelling mechanism to generalise data with uncertain and vague information into mode fragments [37].
– Fuzzy Iterative Inference offers a combination of abductive and deductive inferences, capable of reasoning with uncertain assumptions [15].

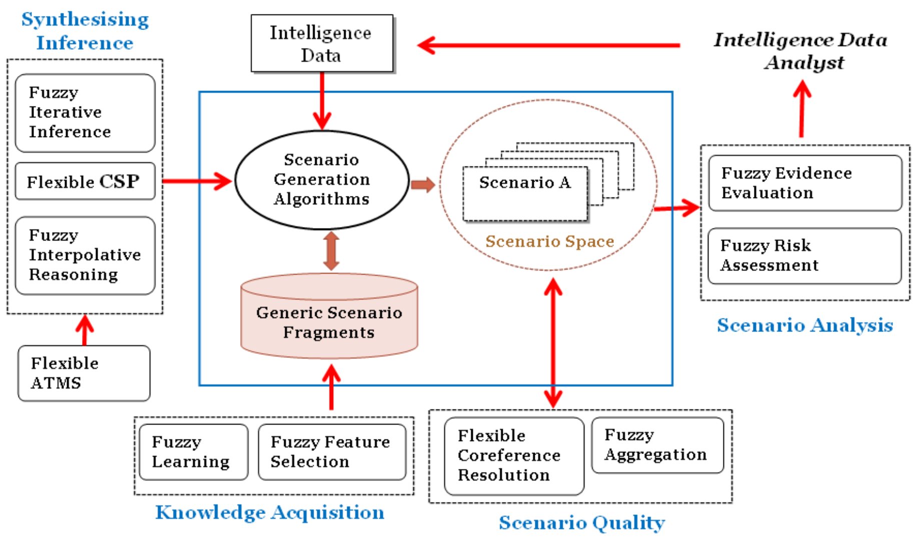


Fig. 2. A Specific Implementation of the Framework

- Flexible CSP (constraint satisfaction problem-solver) deals with uncertain and imprecise constraint satisfaction, subject to preference and priority [38].
- Fuzzy Interpolation enables approximate inference over sparse knowledge base [19].
- Flexible ATMS is an extended truth-maintenance system that keeps track of uncertain assumption-based deduction [51].
- Flexible Coreference Resolution implements a link-based identity resolution approach, working with real, nominal, and order-of-magnitude valued attributes [7].
- Fuzzy Aggregation performs information aggregation by combining uncertain attributes as well as their values [6].
- Fuzzy Evidence Evaluation performs evidence assessment, including discovery of misleading information, and generates evidence-gathering proposal [27].
- Fuzzy Risk Assessment computes potential loss-oriented risk evaluation through fuzzy random process modelling [50].

This research focusses on the use of structured knowledge for modelling and analysis of intelligence data which may contain factual evidence and assumed information. Such data may be imprecise and uncertain, involving vague or even ill-defined concepts. For simplicity, it is assumed that the data is presented in a pre-specified form, e.g. in terms of properties of suspects, types of incident and classes of evidence. This implies that for real application, a preprocessing stage of raw information is generally required to ensure the uniform representation of the data.

Recent advances in semantic web research [3,45] provide useful techniques for preprocessing raw data to reveal the content of source information. The simplest whilst very effective way is to inflate each datum structure with dummy entries for any additional feature that is used in other data. Use of such techniques will help to automate the preprocessing of raw information and to harmonise data representation. Whilst details of such preliminary data handling methods are beyond scope of this work, an investigation of how model fragments may be learned from data that is typically high-dimensional is obviously essential.

Systems built following the approach outlined in Fig. 2 can help to improve the likelihood of discovering any potential threat posed by criminal or terrorist organisations. In particular, the use of an automated intelligent data analysis system, whose reasoning is logical and readily interpretable by human analysts, can be very helpful in supporting human analysts when working under time constraints. For instance, this may aid in avoiding premature commitment to certain seemingly more likely but unreal scenarios, minimising the risk of producing incorrect interpretations of intelligence data. This may be of particular interest to support staff investigating cases with unfamiliar evidence. In addition, the resulting approach may be adapted to build systems that facilitate training of new intelligence analysts. This is possible because the underlying inference mechanism and the knowledge base built for intelligence data analysis can be used to artificially synthesise various scenarios (of whatever likelihood), and to systematically examine the implications of acquiring different types of observation.

The next section describes representative techniques that can be utilised to implement a number of important component systems contained within the specification of the framework (shown in Fig. 2).

3 Component Techniques

As a knowledge-based approach to building systems for intelligent data analysis, any actual implementation of the framework proposed above will require a knowledge base to begin with. The first part of this section will then introduce a number of recent advances in developing data-driven learning techniques that are suitable to derive such required knowledge from potentially very complex data. The second part will describe one of the key techniques that support scenario composition, especially for situations where limited domain knowledge is available. The third and final part of the section will demonstrate how risks of generated scenarios may be estimated such that preferential treatments to the likely events can be made. Figure 3 outlines a simplified version of the framework which may be implemented using the techniques described here.

All of these approaches have been developed using computational intelligence techniques in general and fuzzy systems methods in particular. Introduction to these techniques will be explained at conceptual level with illustrative examples. Mathematical and computational details are omitted, but readers may find them from the relevant references.

3.1 Fuzzy Learning and Feature Selection

In general, an initial knowledge base of generic scenario fragments is built partly by generalising historical intelligence data through computer-based induction, but partly through manual analysis of past terrorist or criminal activity. This work focusses on the automated induction of model fragments. As indicated previously, in a real world

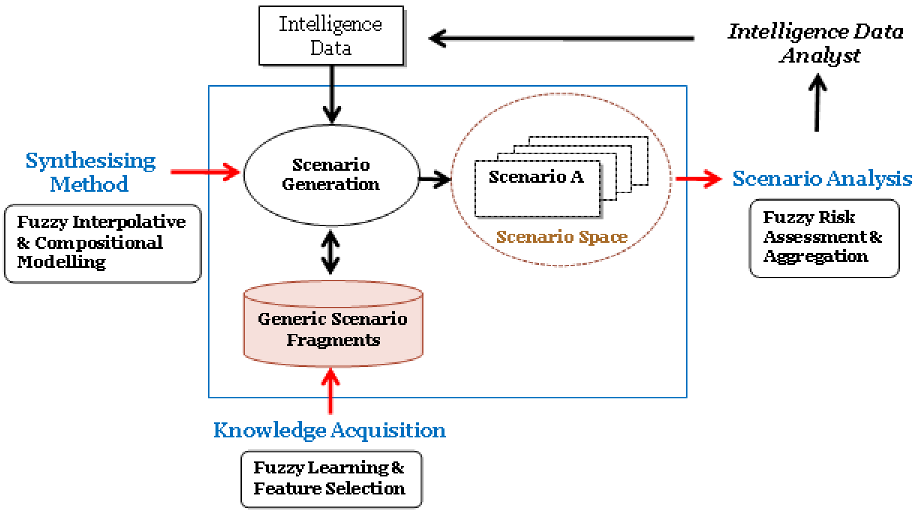

Intelligence Data
Intelligence Data Analyst
Synthesising Method
Fuzzy Interpolative & Compositional Modelling
Scenario Generation
Scenario A
Scenario Space
Scenario Analysis
Fuzzy Risk Assessment & Aggregation
Generic Scenario Fragments
Knowledge Acquisition
Fuzzy Learning & Feature Selection

Fig. 3. Focussed Implementation

setting, data may come from multiple sources and hence, may require a substantial amount of preprocessing in order to achieve semantic interpretability. However, this consideration is beyond the scope of this paper. Any data given for learning is assumed to have been presented in a homogeneous format.

Fuzzy Descriptive Learning. Many real-world problems require the development and application of algorithms that automatically generate human interpretable knowledge from historical data. Such a task is clearly not just for learning model fragments.

Most of the methods for fuzzy rule induction from data have followed the so-called precise approach. Interpretability is often sacrificed, in exchange for a perceived increase in precision. In many cases, the definitions of the fuzzy sets that are intended to capture certain vague concepts are allowed to be modified such that they fit the data better. This modification comes at the cost of ruining the original meaning of the fuzzy sets and the loss of transparency of the resulting model. In other cases, the algorithms themselves generate the fuzzy sets, and present them to the user. The user must then interpret these sets and the rules which employ them (e.g. a rule like: If *volume* is Tri(32.41, 38.12, 49.18), then *chance* is Tri(0.22, 0.45, 0.78)). Furthermore, in some extreme cases, each rule may have its own fuzzy set definition for every condition, thereby generating many different sets in a modest rule base. The greatest disadvantage of the precise approach is that the resulting sets and rules are difficult to match with human interpretation of the relevant concepts.

As an alternative to the precise approach, there exist proposals that follow the descriptive (or linguistic) approach. In such work no changes are made to human defined fuzzy sets. The rules must use the (fuzzy) words provided by the user without modifying them in any way. One of the main difficulties with this type of approach is that the possible rules available are predetermined, equivalently speaking. This is because the

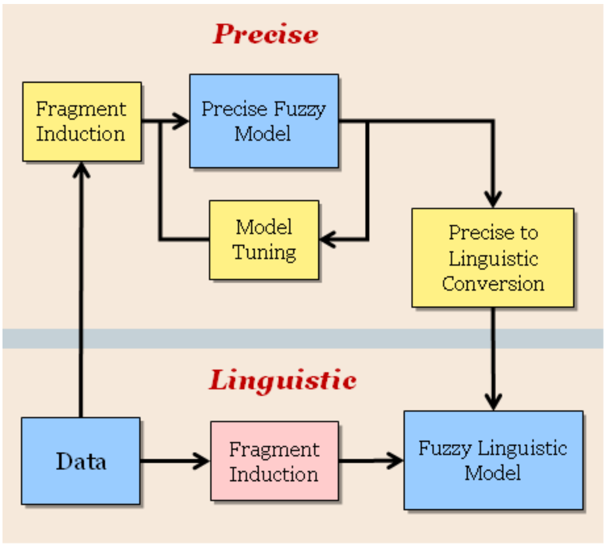


Fig. 4. Two-Step Learning of Descriptive Models

fuzzy sets can not be modified, and only a small number of them are typically available. Although there can be many of these rules they are not very flexible and in many cases they may not necessarily fit the data well (e.g. a rule like: If *volume* is Moderate, then *chance* is High). In order to address this problem, or at least partially, linguistic hedges (aka. fuzzy quantifiers) can be employed.

The concept of hedges has been proposed quite early on in fuzzy systems research [60]. A linguistic hedge produces a new fuzzy set by changing the original fuzzy set, in a fixed and interpretable manner. The interpretation of the resultant set emanates from the original fuzzy set and a specific transformation that the hedge implies. In so doing, the original fuzzy sets are not changed, but the hedged fuzzy sets provide modifiable means of modelling a given problem and therefore, more freedom in representing knowledge in the domain.

This research adopts the original work of [37] which champions this approach. As shown in Fig. 4, this technique produces descriptive fuzzy system models with a two-step mechanism. The first is to use a precise method to create accurate rules and the second to convert the resulting precise rules to descriptive ones. The conversion is, in general, one-to-many. It is implemented by using a heuristic algorithm that derives potentially useful translations and then, by employing evolutionary computation to perform a fine tuning of these translations. Both steps are computationally efficient. The resultant descriptive model (e.g. a rule like: If *volume* is Fairly Moderate, then *chance* is Very High) is ready to be directly applied for inference; no precise rules are needed in runtime.

Note that Fig. 4 shows the learning of a "model" in a general sense. Such a model may be a set of conventional production fuzzy if-then rules, or one or more generic model fragments which involve not only standard conditions but also assumptions or hypotheses that must be made in order to draw conclusions.

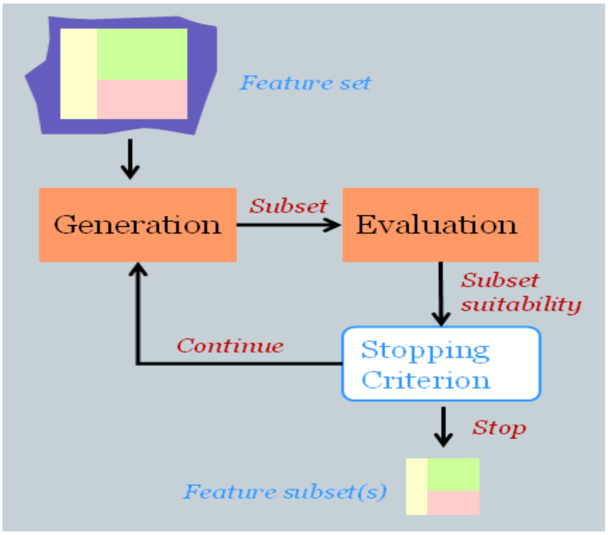


Fig. 5. Feature Selection Process

Fuzzy-rough Feature Selection. Feature selection [23,34] addresses the problem of selecting those characteristic descriptors of a domain that are most informative. Figure 5 shows the basic procedures involved in a feature selection process. It is a problem encountered in many areas of computational intelligence. Unlike other dimensionality-reduction methods, feature selectors preserve the original meaning of the features after reduction. This has been applied to perform tasks that involve datasets containing huge numbers of features (in the order of tens of thousands) which, for some learning algorithms, may be otherwise impossible to process further.

There are often many features involved in intelligence data, and combinatorially large numbers of feature combinations, to select from. It might be expected that the inclusion of an increasing number of features would increase the likelihood of including enough information to distinguish between classes. Unfortunately, this is not necessarily true if the size of the training dataset does not also increase rapidly with each additional feature included. A high-dimensional dataset increases the chances that a learning algorithm will find spurious patterns that are not valid in general. More features may introduce more measurement noise and, hence, reduce model accuracy [21].

There have been significant advances in developing methodologies that are capable of minimising feature subsets in an imprecise and uncertain environment. Sophisticated approaches exist which attempt to identify or approximate the absolute smallest subsets of features. However, in general, searching for all minimal feature subsets is an NP-complete problem. In practice, it may suffice to generate only one such subset or even a superset of some of such subsets (if it is too time consuming to compute a global minimal).

A resounding amount of recent research utilises fuzzy and rough sets (e.g. [35,36,48,54]), following the seminal work of [11,47]. The success of rough set theory [42] is due in part to the following two aspects: (a) only the facts hidden in data are analyzed, and (b) no additional information about the data is required, such as thresholds or expert knowledge on a particular domain. However, it handles only one type of imperfection found in data, it is complementary to other mathematical concepts for this purpose, e.g. fuzzy set theory. In fact, the rough and fuzzy fields may be considered analogous in the sense that both can tolerate inconsistency and uncertainty [25]. The difference rests in the type of uncertainty and their approach to it; fuzzy sets are concerned with vagueness, and rough sets are concerned with indiscernibility. Therefore, it is desirable to extend and hybridise the underlying concepts to deal with additional aspects of data imperfection [14].

A particular representative of such work is fuzzy-rough feature selection [22,24]. It provides a means by which discrete or real-valued noisy data (or a mixture of both) can be effectively reduced without the need for user-supplied information. Importantly, this technique can be applied to data with continuous or nominal decision attributes, and as such is suitable for the nature of intelligence data. A particular implementation is done via hill-climbing search, as shown in Algorithm 1. What is returned by this algorithm is the subset of features selected from the full set of original features without altering the meaning and value of such selected features.

Algorithm 1: The Fuzzy-Rough Feature Selection (FRFS) Algorithm

```
FRFS($C, D$): $C$ – the set of all conditional features; $D$ – the set of decision features.

(1)  $R \leftarrow \{\}, \gamma_{best} = 0$
(2)  do
(3)      $T \leftarrow R, \gamma_{prev} \leftarrow \gamma_{best}$
(4)      foreach $a \in (C - R)$
(5)          if $\gamma_{R \cup \{a\}}(D) > \gamma_T(D)$
(6)              $T \leftarrow R \cup \{a\}, \gamma_{best} \leftarrow \gamma_T D$
(7)      $R \leftarrow T$
(8)  until $\gamma_{best} == \gamma_{prev}$
(9)  return $R$
```

This algorithm employs the fuzzy-rough dependency function, which is derived from the notion of fuzzy lower approximation, to choose those attributes that add to the current candidate feature subset in a best-first fashion. The algorithm terminates when the addition of any remaining attribute does not result in an increase in the dependency. As such, all features selected are individually significant, although significance embedded in any correlated features may not be necessarily captured. This implies that the feature subsets returned may not be globally minimal. Yet, this general problem is due to the use of greedy hill-climbing search, not because of the utilisation of fuzzy-rough dependency measure.

3.2 Fuzzy Interpolation and Extrapolation

In conventional approaches to compositional modelling, the completeness of a scenario space depends upon two factors: (a) the knowledge base must cover all essential

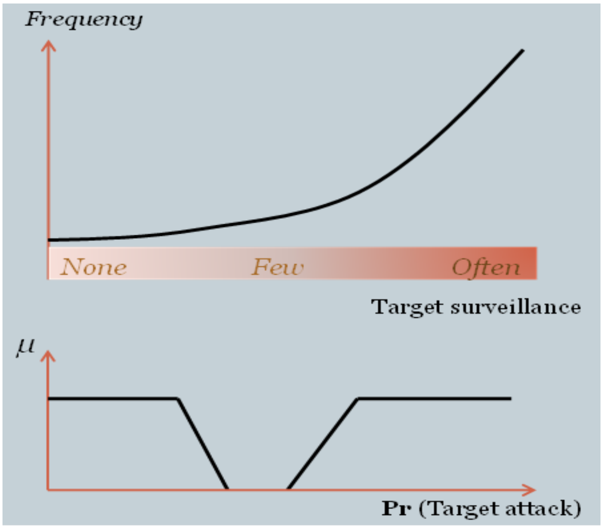


Fig. 6. Need for Fuzzy Interpolation

scenario fragments relevant to the data, and (b) the inference mechanism must be able to synthesise and store all combinations of instances of such fragments that constitute a consistent scenario. However, in the real-world, especially for the problem domain concerned here, it is difficult, if not impossible, to obtain a complete library of model fragments. Figure 6 shows an example, where the following two simplified model fragments (i.e. two simple if-then rules in this case) are given:

$Rule_i$: If *frequency* is None then *attack* is No
$Rule_j$: If *frequency* is Often then *attack* is Yes

Then, with an observation that states "*frequency* is Few", no answer can be found to the question of "Will there be an attack"? A popular tool to deal with this type of problem is fuzzy interpretative reasoning [2,53]. In this work, the transformation-based approach [19,20] is employed to support model composition, when given an initial sparse knowledge base.

The need for a fuzzy approach to interpolation is due to the fact that the precision degree of the available intelligence data can be variable. Finding a match between the given data and the (already sparse) knowledge base cannot in general be achieved precisely. The potential sources of variability in precision include vaguely defined concepts (e.g. materials that constitute a "high explosive", certain organisations that are deemed "extremist"), quantities (e.g. a "substantial" amount of explosives, "many" people) and specifications of importance and certainty (e.g. in order to deploy a radiological dispersal device, the perpetrator "must" have access to "radioactive material and "should" have an ideological or financial incentive). Therefore, the approach adopted must be able to describe and reason with knowledge and data at varying degrees of precision. Fuzzy interpolation works well in this regard.

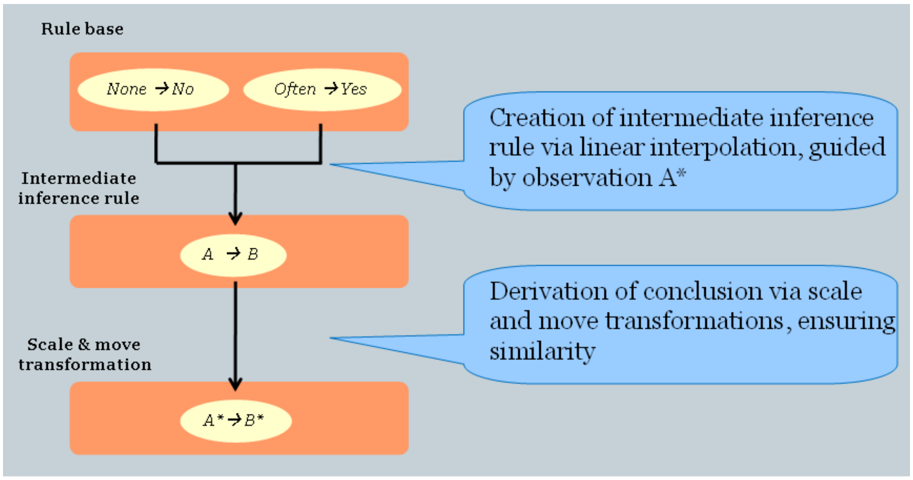

Rule base
None → No
Often → Yes
Creation of intermediate inference rule via linear interpolation, guided by observation A*
Intermediate inference rule
A → B
Derivation of conclusion via scale and move transformations, ensuring similarity
Scale & move transformation
A* → B*

Fig. 7. Transformation-Based Fuzzy Interpolation

Figure 7 illustrates the basic procedure of fuzzy interpolation. It works through a two-step process: (a) computationally constructing a new inference rule (or model fragment in the present context) via manipulating two given adjacent rules (or related fragments), and (b) using scale and move transformations to convert the intermediate inference results into the final derived conclusions.

Although the illustrative example only involves interpolation with two rules which are each of one conditional feature, the underlying approach is more general, covering extrapolation as well as interpolation with multiple rules involving conditional features. Work has also been done in automated correction of errors incurred during the interpolation/extrapolation process due to inaccurate data [58].

3.3 Fuzzy Risk Assessment

Serious crime may cause considerable loss of life and damage to property. For example, the terrorist attack on the World Trade Center on September 11, 2001 claimed around 3000 lives [4] and caused an estimated "$120 billion of damage" [55]. Rapid and accurate estimation of the risk of any such potential crime will help to provide a useful means for the establishment of appropriate risk management and loss mitigation strategies. It is therefore crucial to develop reliable methods for risk assessment of serious crime events using available intelligence data.

In developing intelligent systems for intelligence data modelling and analysis, there is often a trade-off that must be considered. That is, between the completeness of the scenario space generated and the potential efficiency in subsequent examination of the resultant space. On the one hand, it is important not to miss any potentially significant scenarios that may explain the observed data. On the other hand, too many unsorted and especially, spurious scenarios produced may confuse human analysts. Thus, it is desirable to be able to filter the created scenario space with respect to certain objective measures of the quality of the generated scenario descriptions. Fortunately, as indicated previously, preferences over different hypothetical scenarios can be determined on the basis of reliability, risk and urgency of each scenario.

The *reliability* of a generated scenario may be affected by several distinct factors: the given intelligence data (e.g. the reliability of an informant), the inferences made to abduce plausible scenarios (e.g. the probability that a given money transfer is part of an illegitimate transaction), and the default assumptions adopted (e.g. the likelihood that a person seen on CCTV footage is identified positively). The *urgency* of a scenario is inversely proportional to the expected time to completion of a particular terrorist/criminal activity. Therefore, an assessment of urgency requires a (partial) scenario to be described using the scenario's possible consequences and information on additional actions required to achieve completion. The *risk* posed by a particular scenario is determined by its potential consequences (e.g. damage to people and property). Whilst these are very different aspects that may be used to differentiate and prioritise scenarios composed by the compositional modeller, the underlying approaches to assess them are very similar. Thus, in this paper, only the scenario risk aspect is discussed.

Various approaches have been proposed for risk assessment of crime. In [56], crime risk involves three components (in terms of terrorism): threat to a target, target vulnerability, and the consequence should the target be successfully attacked (from the terrorist's viewpoint). It is abstractly defined as the product of these three factors (all of which are regarded as random variables). Statistical methods are employed to estimate their probability distributions and expected values. An alternative approach is given in [12], which uses possibility theory to address the general issue of uncertainty. In particular, concepts such as attack, success, and consequence are characterized as discrete fuzzy sets and crime risk is then defined as the convolution of them. The risk of serious crime may also be viewed as an "extreme value event" as in [39]. Thus, it can be measured by using extreme value statistics. In addition, loss analysis caused by serious crime may be carried out with methods drawn from fields such as game theory, social psychology, and network analysis [57]. Clearly, the models of crime risk vary from one approach to another, depending on a variety of issues, including: type of crime, scale of crime, and the uncertainty of crime. Of course, risk analysis and assessment is not limited to the area of crime detection and prevention. It is a general problem that needs to be addressed in many different domains [8,10,18].

Estimating the risk of a serious crime event requires consideration of a large amount of uncertainty due to both randomness and fuzziness that are inherent in the intelligence data (and knowledge also). In this work, the occurrence of crime activity is considered as a random event, while the loss incurred by the crime is considered as a fuzzy value. The proposed approach adopts fuzzy random theory [30,31,40] to cope with the challenge of assessing crime risk and to support loss analysis. In particular, risk is estimated as the mean chance of a fuzzy random event [17,52] over a pre-defined confidence level, for an individual type of loss. In implementation, loss caused by an event is modelled by a function mapping from a boolean sample space of $\{$Success, Failure$\}$ onto a set of nonnegative fuzzy variables. Here, success or failure is judged from the criminal's viewpoint, of course, in terms of whether they have carried out a certain activity or not.

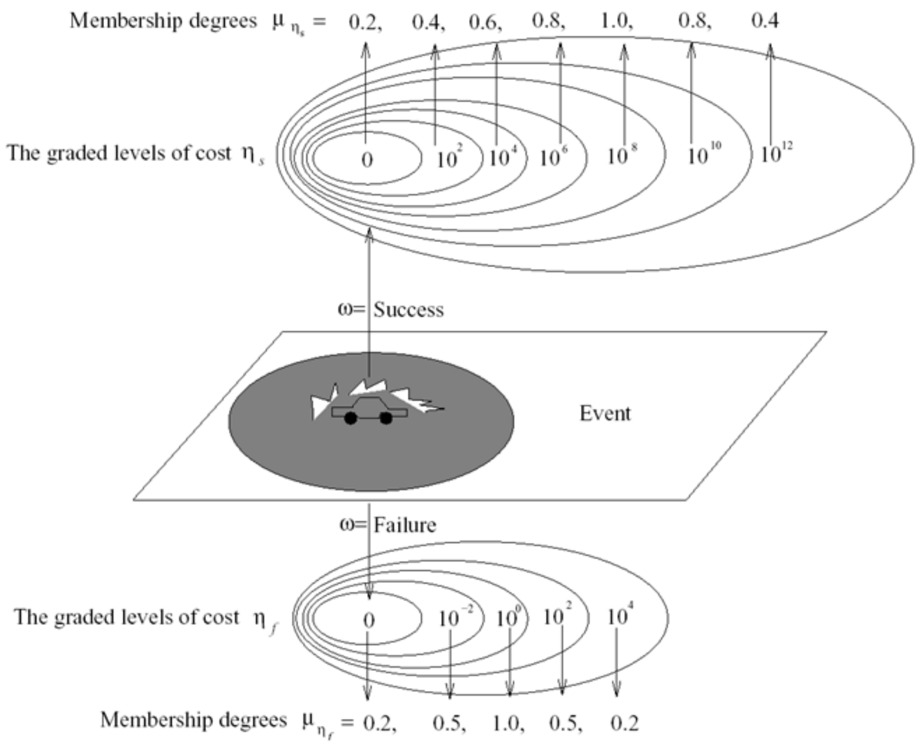

Membership degrees μ_{η_s} = 0.2, 0.4, 0.6, 0.8, 1.0, 0.8, 0.4
The graded levels of cost η_s
0
10^2
10^4
10^6
10^8
10^{10}
10^{12}
ω= Success
Event
ω= Failure
The graded levels of cost η_f
0
10^{-2}
10^0
10^2
10^4
Membership degrees μ_{η_f} = 0.2, 0.5, 1.0, 0.5, 0.2

Fig. 8. Risk Assessment

Risks estimated over different types of loss (e.g. range of geometric destruction and number of casualties) can be aggregated. Also, assessments obtained using different criteria (e.g. resource and urgency) may be integrated to form an overall situation risk. To generalise this approach further, order-of-magnitude representation [41,44] may be introduced to describe various cost estimations. Figure 8 shows such an example.

Incidentally, expert intelligence analysts have commented that the estimation in this figure matches the real dataset used in the corresponding set of terrorist attacks. However, they also commented that in general, it is not necessarily always the case that failure in carrying out a certain terrorist attack (or the successful prevention of possible terrorist attack from the counter-terrorism perspective) would incur lower cost to the public. The ability of an intelligence data analysis tool to correctly capture such cases clearly deserves more through investigation in future. A possible approach to addressing this issue is to utilise the measures of risk, urgency and reliability as flexible constraints imposed over the planning process of police resource deployment. This will help to minimise the cost of successful surveillance, for example. Techniques reported in [38] may be used to automate such planning.

4 Conclusions

This paper has presented an initial framework upon which to develop intelligent systems for modelling and analysis of intelligence data. It has proposed methods which can aid intelligence analysts in considering as widely as possible the range of emerging scenarios that may reflect serious criminal activities. The resulting approach has the ability to link seemingly distinct and unrelated intelligence data, associating and prioritising such data with logically inferred and justified hypotheses.

In short, this work has demonstrated that computational intelligence in general, and fuzzy systems in particular can provide useful means to capture, learn and reason from intelligence data under uncertainty. This research has reflected the fact that fuzzy techniques can be very successful for:

- Fragment induction
- Feature selection
- Data interpolation
- Model composition
- Constraint satisfaction
- Truth maintenance
- Co-reference resolution
- Information aggregation
- Risk assessment

Whilst this research is very promising, important work remains. In addition to what has been mentioned in the paper, the following lists a number of further issues (amongst possibly many others) that are worthy of investigation and/or development in order to reinforce the potential of this initial investigation:

- Learning hierarchical model fragments
- Hierarchical and ensemble feature selection
- Unification of scenario generation algorithms
- Dynamic coreference resolution and information fusion
- Evidence-driven risk-guided scenario generation
- Discovery of rare and misleading cases
- Meta-feature learning and selection for scenario synthesis

Such further studies may also bring up fresh challenges to computational intelligence research and hence new technologies for building intelligent data analysis systems. It will help to consolidate and broaden the application scope of the proposed framework. In particular, the framework may be adapted to tackle problems such as: investigator training, policy formulation, multi-modal profiling, and to address issues in domains such as academic performance evaluation and financial situation forecasting.

References

1. Aitken, C., Shen, Q., Jensen, R., Hayes, B.: The evaluation of evidence for exponentially distributed data. Computational Statistics and Data Analysis 51, 5682–5693 (2007)
2. Baranyi, P., Koczy, L., Gedeon, T.: A generalized concept for in fuzzy rule interpolation. IEEE Transactions on Fuzzy Systems 12(6), 820–837 (2004)
3. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American, 34–43 (May 2001)
4. CNN: A Day of Terror (September 11, 2001), http://www.cnn.com/2003/US/03/10/sprj.80.2001.terror/index.html
5. Berthold, M., Hand, D.: Intelligent Data Analysis: An Introduction, 2nd edn. Springer, Heidelberg (2007)
6. Boongoen, T., Shen, Q.: Nearest-neighbor guided evaluation of data reliability and its applications. IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics (to appear)
7. Boongoen, T., Shen, Q., Price, P.: Disclosing false identity through hybrid link analysis. Artificial Intelligence and Law 18(1), 77–102 (2010)
8. Buyukozkan, G., Ruan, D.: Choquet integral based aggregation approach to software development risk assessment. Information Sciences 180, 441–451 (2010)
9. Calado, P., Cristo, M., Goncalves, M., de Moura, E., Ribeiro-Neto, E., Ziviani, N.: Link based similarity measures for the classification of web documents. Journal of American Society fo Information Science and Technology 57(2), 208–221 (2006)
10. Chen, S., Huang, Y.: Relative risk aversion and wealth dynamics. Information Sciences 177, 1222–1229 (2007)
11. Chouchoulas, A., Shen, Q.: Rough set-aided keyword reduction for text categorisation. Applied Artificial Intelligence 15(9), 843–873 (2001)
12. Darby J.: Estimating terrorist risk with possibility theory (2004), http://www.doe.gov/bridge
13. de Kleer, J.: An assumption-based TMS. Artificial Intelligence 28(2), 127–162 (1986)
14. Dubois, D., Prade, H.: Rough fuzzy sets and fuzzy rough sets. International Journal of General Systems 17, 191–209 (1990)
15. Fu, X., Boongoen, T., Shen, Q.: Evidence directed generation of plausible crime scenarios with identity resolution. Applied Artificial Intelligence 24(4), 253–276 (2010)
16. Fu, X., Shen, Q.: Fuzzy compositional modeling. IEEE Transactions on Fuzzy Systems 18(4), 823–840 (2010)
17. Halliwell, J., Shen, Q.: Linguistic probabilities: theory and application. Soft Computing 13(2), 169–183 (2009)
18. Huang, C., Inoue, H.: Soft risk maps of natural disasters and their applications to decision-making. Information Sciences 177, 1583–1592 (2007)
19. Huang, Z., Shen, Q.: Fuzzy interpolative and extrapolative reasoning: a practical approach. IEEE Transactions on Fuzzy Systems 16(1), 13–28 (2008)
20. Huang, Z., Shen, Q.: Fuzzy interpolative reasoning via scale and move transformation. IEEE Transactions on Fuzzy Systems 14(2), 340–359 (2006)
21. Jensen, R., Shen, Q.: Are more features better? IEEE Transactions on Fuzzy Systems 17(6), 1456–1458 (2009)
22. Jensen, R., Shen, Q.: New approaches to fuzzy-rough feature selection. IEEE Transactions on Fuzzy Systems 17(4), 824–838 (2009)
23. Jensen, R., Shen, Q.: Computational Intelligence and Feature Selection: Rough and Fuzzy Approaches. IEEE and Wiley (2008)

24. Jensen, R., Shen, Q.: Fuzzy-rough sets assisted attribute selection. IEEE Transactions on Fuzzy Systems 15(1), 73–89 (2007)
25. Jensen, R., Shen, Q.: Semantics-preserving dimensionality reduction: Rough and fuzzy-rough approaches. IEEE Transactions on Knowledge and Data Engineering 16(12), 1457–1471 (2004)
26. Keppens, J., Shen, Q.: On compositional modelling. Knowledge Engineering Review 16(2), 157–200 (2001)
27. Keppens, J., Shen, Q., Price, C.: Compositional Bayesian modelling for computation of evidence collection strategies. Applied Intelligence (to appear)
28. King, R., Rowland, J., Oliver, S., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L., Sparkes, A., Whelan, K.E., Clare, A.: The automation of science. Science 324(5923), 85–89 (2009)
29. Koyuncu, M., Yazici, A.: A fuzzy knowledge-based system for intelligent retrieval. IEEE Transactions on Fuzzy Systems 13(3), 317–330 (2005)
30. Kwakernaak, H.: Fuzzy random variables – I. Information Sciences 15, 1–29 (1978)
31. Kwakernaak, H.: Fuzzy random variables – II. Information Sciences 17, 253–278 (1979)
32. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of American Society for Information Science and Technology 58(7), 1019–1031 (2007)
33. Kriete, A., Eils, R.: Computational Systems Biology. Elsevier, Amsterdam (2005)
34. Liu, H., Motoda, H.: Feature Selection for Knowledge Discovery and Data Mining. Springer, Heidelberg (1998)
35. Mac Parthalain, N., Shen, Q.: Exploring the boundary region of tolerance rough sets for feature selection. Pattern Recognition 42(5), 655–667 (2009)
36. Mac Parthalain, N., Shen, Q., Jensen, R.: A distance measure approach to exploring the rough set boundary region for attribute reduction. IEEE Transactions on Knowledge and Data Engineering (to appear)
37. Marín-Blázquez, J., Shen, Q.: From approximative to descriptive fuzzy classifiers. IEEE Transactions on Fuzzy Systems 10(4), 484–497 (2002)
38. Miguel, I., Shen, Q.: Fuzzy rrDFCSP and planning. Artificial Intelligence 148(1-2), 11–52 (2003)
39. Mohtadi, H.: Assessing the risk of terrorism using extreme value statistics. In: Proceedings of the Institute of Food Technologists First Annual Conference on Food Protection and Defencse (2005)
40. Puri, M., Ralescu, D.: Fuzzy random variables. Journal of Mathematical Analysis and Applications 114, 409–422 (1986)
41. Parsons, S.: Qualitative probability and order of magnitude reasoning. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 11(3), 373–390 (2003)
42. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishers, Dordrecht (1991)
43. Perner, P.: Intelligent data analysis in medicine: Recent advances. Artificial Intelligence in Medicine 37(1), 1–5 (2006)
44. Raiman, O.: Order-of-magnitude reasoning. Artificial Intelligence 51, 11–38 (1991)
45. Shadbolt, N., Hall, W., Berners-Lee, T.: The semantic web revisited. IEEE Intelligent Systems 21(3), 96–101 (2006)
46. Shen, Q.: Intelligent systems for decision support. In: Proceedings of International Joint Conference on Computational Intelligence, pp. 25–36 (2009)
47. Shen, Q., Chouchoulas, A.: A rough-fuzzy approach for generating classification rules. Pattern Recognition 35(11), 2425–2438 (2002)
48. Shen, Q., Jensen, R.: Selecting informative features with fuzzy-rough sets and its application for complex systems monitoring. Pattern Recognition 37(7), 1351–1363 (2004)

49. Shen, Q., Keppens, J., Aitken, C., Schafer, B., Lee, M.: A scenario driven decision support system for serious crime investigation. Law, Probability and Risk 5(2), 87–117 (2006)
50. Shen, Q., Zhao, R.: Risk assessment of serious crime with fuzzy random theory. Information Sciences (to appear)
51. Shen, Q., Zhao, R.: A credibilistic approach to assumption-based truth maintenance. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans (to appear)
52. Shen, Q., Zhao, R., Tang, W.: Modelling random fuzzy renewal reward processes. IEEE Transactions on Fuzzy Systems 16(5), 1379–1385 (2008)
53. Tikk, D., Baranyi, P.: Comprehensive analysis of a new fuzzy rule interpolation method. IEEE Transactions on Fuzzy Systems 8(3), 281–296 (2000)
54. Tsang, E., Chen, D., Yeung, D., Wang, X., Lee, J.: Attributes reduction using fuzzy rough sets. IEEE Transactions on Fuzzy Systems 16(5), 1130–1141 (2008)
55. Wesbury B.: The Economic Cost of Terrorism, http://usinfo.state.gov/topical/econ/mlc/02091004.htm
56. Willis H., Morral A., Kelly T., Medby J.: Estimating terrorism risk, RAND Corporation, Report from Center for Terrorism Risk Management Policy (2005), http://www.rand.org
57. Woo, G.: Terrorism risk. In: Voeller, J. (ed.) Handbook of Science and Technology for Homeland Security. Wiley, Chichester (2007)
58. Yang, L., Shen, Q.: Adaptive fuzzy interpolation. IEEE Transactions on Fuzzy Systems (to appear)
59. Zadeh, L.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems 100, 9–34 (1999)
60. Zadeh, L.: The concept of a linguistic variable and its application to approximate reasoning I. Information Sciences 8, 199–249 (1975)
61. Zadeh, L.: Fuzzy sets. Information and Control 8(3), 338–353 (1965)

Part I
Fuzzy Computation

Symbolic Knowledge Extraction from Trained Neural Networks Governed by Łukasiewicz Logics[*]

Carlos Leandro[1], Hélder Pita[2], and Luís Monteiro[3]

[1] Área Cientifica da Matemática, Instituto Superior de Engenharia de Lisboa
Instituto Politécnico de Lisboa, Portugal
[2] Departamento de Engenharia Electrónica Telecomunicaçes e de Computadores
Instituto Superior de Engenharia de Lisboa, Instituto Politécnico de Lisboa, Portugal
[3] Departamento de Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, Portugal

Abstract. This work describes a methodology to extract symbolic rules from trained neural networks. In our approach, patterns on the network are codified using formulas on a Łukasiewicz logic. For this we take advantage of the fact that every connective in this multi-valued logic can be evaluated by a neuron in an artificial network having, by activation function the identity truncated to zero and one. This fact simplifies symbolic rule extraction and allows the easy injection of formulas into a network architecture. We trained this type of neural network using a back-propagation algorithm based on Levenderg-Marquardt algorithm, where in each learning iteration, we restricted the knowledge dissemination in the network structure. This makes the descriptive power of produced neural networks similar to the descriptive power of Łukasiewicz logic language, minimizing the information loss on the translation between connectionist and symbolic structures. To avoid redundance on the generated network, the method simplifies them in a pruning phase, using the "Optimal Brain Surgeon" algorithm. We tested this method on the task of finding the formula used on the generation of a given truth table. For real data tests, we selected the *Mushrooms* data set, available on the *UCI Machine Learning Repository*.

1 Introduction

There are essentially two representation paradigms, symbolic-based descriptions and connectionist, usually taken very differently. On the one hand, symbolic-based descriptions are based on a language specified through a grammar that has fairly clear semantics. On the other hand, the usual way to see information presented using a connectionist description is its codification on a neural network. Artificial neural networks (NNs), in principle, combine–among other things–the ability to learn with robustness or insensitivity to perturbations of input data.

It is natural to seek a synergy integrating the *white-box* character of symbolic based representation and the learning power of artificial neuronal networks. Such neuro-symbolic models are currently a very active area of research. In the context of classic logic,

[*] We gratefully acknowledge the support of the Instituto Superior de Engenharia de Lisboa and the Área Cientifica da Matemática. This paper benefited from the valuable comments of ICFC2009 conference referees.

see [1] for the extraction of logic programs from trained networks. For the extraction of modal and temporal logic programs, see [2]. In [3] we can find processes to generate connectionist representations of multi-valued logic programs and for Łukasiewicz logic programs (ŁL) [4]. Our approach to neuro-symbolic models and knowledge extraction is based on a comprehensive language for humans, representable directly in an NN topology and able to be used, like knowledge-based networks [5] [6], to generate the initial network architecture from crude symbolic domain knowledge. In the other direction, neural language can be translated into its symbolic language, as presented in [7] [8]. However this process has been used to identify the most significant determinants of decision or classification. This is a hard problem since often an artificial NN with good generalization does not necessarily imply involvement of hidden units with distinct meaning. Hence, any individual unit cannot essentially be associated with a single concept or feature of the problem domain. This is the archetype of connectionist approaches, where all information is stored in a distributed manner among the processing units and their associated connectivity. However, in this work we used a propositional language wherein formulas are interpreted as NNs. For this task we selected the propositional language of ŁL. This type of multi-valued logic has a very useful property motivated by the "linearity" of logic connectives. Every logic connective can be defined by a neuron in an artificial network having, by activation function, the identity truncated to zero and one [9]. This allows the direct codification of formulas in the network architecture and simplifies the extraction of rules. Multilayer feed-forward NN, having this type of activation function, can be trained efficiently using the Levenderg-Marquardt algorithm [10], and the generated network can be simplified using the "Optimal Brain Surgeon" algorithm proposed by B. Hassibi, D. G. Stork, and G. J. Stork [11]. This strategy has good performance when applied to the reconstruction of formulas from truth tables. If the truth table is generated using a formula from the Łukasiewicz propositional logic language, the optimum solution is defined using only units directly translated into formulas. The process is stable for the introduction of Gaussian noise into the input data. This motivates its application to extract comprehensible symbolic rules from real data. However, often a model with good generalization can be described using a configuration of neural units without exact symbolic presentation. We describe, in the following, a simple rule to generate symbolic approximation for un-representable configurations.

2 Preliminaries

We begin by presenting the basic notions we need from the subjects of many-valued logics and by showing how formulas in a propositional language can be injected into and extracted from a feed-forward NN.

2.1 Łukasiewicz Logics

Classical propositional logic is one of the earliest formal systems of logic. The algebraic semantics of this logic are given by Boolean algebra. Both the logic and the algebraic semantics have been generalized in many directions [12]. The generalization of Boolean algebra can be based on the relationship between conjunction and implication given by $(x \wedge y) \le z \Leftrightarrow x \le (y \rightarrow z)$. These equivalences, called *residuation equivalences*, imply the properties of logic operators in Boolean algebras.

In applications of fuzzy logic, the properties of the Boolean conjunction are too rigid; hence it is extended with a new binary connective, $\otimes$, which is usually called *fusion*, and the residuation equivalence $(x \otimes y) \le z \Leftrightarrow x \le (y \Rightarrow z)$ defines *implication*. These two operators induce a structure of *residuated poset* on a partially ordered set of truth values P[12]. This structure has been used in the definition of many types of logics. If P has more than two values, the associated logics are called *many-valued logics*.

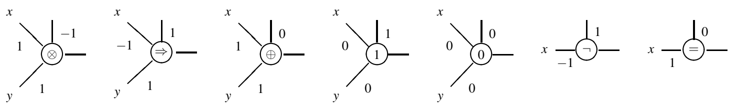
We focused our attention on many-valued logics having a subset of interval $P = [0,1]$ as set of truth values. In this type of logic, the fusion operator, $\otimes$, is known as a t-norm. In [13], it is described as a binary operator defined in $[0,1]$ commutative and associative, non-decreasing in both arguments, $1 \otimes x = x$ and $0 \otimes x = 0$. An example of a continuous t-norm is $x \otimes y = \max(0, x+y-1)$, named as the *Łukasiewicz* t-norm and used on the definition of Łukasiewicz logic (ŁL)[14].

2.2 Processing Units

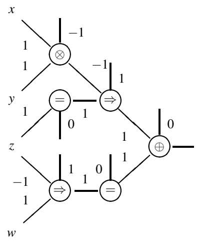
As mentioned in [15] a deep investigation of the relationships between logics and NNs is lacking. In this work we present a methodology using NNs to learn formulas from data, based on the fact [9] that it is possible to represent an NN as a combination of propositions of ŁL, and vice versa [15] when the NN have as activation function, ψ, the identity truncated to zero and one, $\psi(x) = \min(1, \max(x, 0))$.

Sentences in ŁL are, as usual, built from a (countable) set of propositional variables, a conjunction $\otimes$ (the fusion operator), an implication $\Rightarrow$, and the truth constant 0. Further connectives are defined as $\neg\varphi_1$ is $\varphi_1 \Rightarrow 0$, 1 is $0 \Rightarrow 0$ and $\varphi_1 \oplus \varphi_2$ is $\neg\varphi_1 \Rightarrow \varphi_2$.

The language for ŁL can be defined by the set of all multi-layer NNs, with one output neuron, wherein neurons assume one of the labeled configurations presented below[15].



Interpreting each component as a neural unit, each NN can be seen as an interpretation for a formula. A neuron what has bias b and two inputs x_1 and x_2, having weights w_1 and w_2, is interpreted as a function $z = \min(1, \max(0, w_1x + w_2y + b)$. These functions are generically denoted, in the following, by $\psi_b(w_1x_1, w_2x_2)$. In this context a network is the *functional interpretation* for a sentence when relation, defined by network execution, corresponds to the proposition truth table. The use of NNs as the interpretation of formulas simplifies the transformation between string-based representations and the network representation [16]. For instance, the semantic for sentence $\varphi = (x \otimes y \Rightarrow z) \oplus (z \Rightarrow w)$ can be described using the network below, coded by the presented set of matrices. From these matrices we must note that the *partial interpretation* of each unit is a simple exercise of pattern checking, using Table 1, and is described by matrices lines.



$$\begin{array}{cccccl}
 & x & y & z & w & b\text{'s} & \text{partial interpretation} \\
i_1 & 1 & 1 & 0 & 0 & -1 & x \otimes y \\
i_2 & 0 & 0 & 1 & 0 & 0 & z \\
i_3 & 0 & 0 & -1 & 1 & 1 & z \Rightarrow w
\end{array}$$

$$\begin{array}{ccccc l}
 & i_1 & i_2 & i_3 & & \\
j_1 & -1 & 1 & 0 & 1 & i_1 \Rightarrow i_2 \\
j_2 & 0 & 0 & 1 & 0 & i_3
\end{array}$$

$$\begin{array}{cccl}
j_1 & j_2 & & \\
1 & 1 & 0 & j_1 \oplus j_2
\end{array}$$

INTERPRETATION:
$j_1 \oplus j_2 = (i_1 \Rightarrow i_2) \oplus (i_3) = ((x \otimes y) \Rightarrow z) \oplus (z \Rightarrow w)$

In this sense, this NN can be seen as an interpretation for sentence φ; it defines f_φ, the proposition truth table. This relationship is presented in string-base notation by writing: $f_\varphi(x,y,z,w) = \psi_0(\psi_0(\psi_1(-z,w)),\psi_1(\psi_0(z),-\psi_{-1}(x,y)))$.

However, truth table f_φ is a continuous structure for our computational goal, so it must be discretized, ensuring sufficient information to describe the original formula. A truth table f_φ for formula φ, in ŁL, is a map $f_\varphi : [0,1]^m \to [0,1]$, where m is the number of propositional variables used in φ. For each integer $n > 0$, let S_n be the set $\{0, \frac{1}{n}, \ldots, \frac{n-1}{n}, 1\}$. Each $n > 0$, defines a sub-table for f_φ defined by $f_\varphi^{(n)} : (S_n)^m \to [0,1]$, given by $f_\varphi^{(n)}(\bar{v}) = f_\varphi(\bar{v})$, and called the φ *(n+1)-valued truth sub-table*.

2.3 Similarity between a Configuration and a Formula

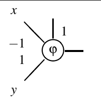
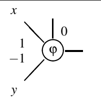
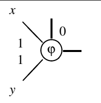
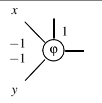
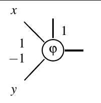
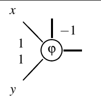
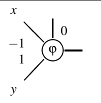
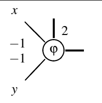
We called a *Castro neural network* (CNN) to an NN having as activation function $\psi(x) = \min(1, max(0,x))$, where its weights are -1, 0 or 1 and having by bias an integer. A CNN is called *representable* if it is codified as a binary NN, i.e., a CNN where each neuron has one or two inputs. A network is called *un-representable* if it is impossible to be codify using a binary CNN. Note that a binary CNN can be translated directly into ŁL language, using the correspondences described in Table 1; in this sense, we called them *Łukasiewicz neural network* (ŁNN). Below we present a functional interpretation for formulas defined using a neuron with two inputs. These interpretations are classified as disjunctive interpretations or conjunctive interpretations. Below we present the possible configurations of neurons with two inputs.

Disjunctive interpretations	Conjunctive interpretations
$\psi_0(x_1,x_2) = f_{x_1 \oplus x_2}$, $\psi_1(-x_1,x_2) = f_{\neg x_1 \oplus x_2}$	$\psi_{-1}(x_1,x_2) = f_{x_1 \otimes x_2}$, $\psi_0(-x_1,x_2) = f_{\neg x_1 \otimes x_2}$
$\psi_1(x_1,-x_2) = f_{x_1 \oplus \neg x_2}$, $\psi_2(-x_1,-x_2) = f_{\neg x_1 \oplus \neg x_2}$	$\psi_0(x_1,-x_2) = f_{x_1 \otimes \neg x_2}$, $\psi_1(-x_1,-x_2) = f_{\neg x_1 \otimes \neg x_2}$

The other possible configurations are equivalent to one of these, or are constant and can also be seen as representable configurations; for instance, $\psi_b(x_1,x_2) = 0$, if $b < -1$, and $\psi_b(-x_1,-x_2) = 1$, if $b > 1$.

In this sense, every representable network can be codified by an NN, where the neural units satisfy one of the above patterns. Below we can also see examples of representable configurations for a neuron with three inputs. In the table we present how they can be codified, using representable NNs having units with two inputs and the corresponding interpreting formula in the string-based notation.

Table 1. Possible configurations for a neuron in a Łukasiewicz neural network and its interpretation

Formula:	Configuration:	Formula:	Configuration:	Formula:	Configuration:	Formula:	Configuration:
$\neg x \oplus y$		$x \otimes \neg y$		$x \oplus y$		$\neg x \otimes \neg y$	
$x \oplus \neg y$		$x \otimes y$		$\neg x \otimes y$		$\neg x \oplus \neg y$	

Conjunctive configurations	Disjunctive interpretations
$\psi_{-2}(x_1,x_2,x_3) = \psi_{-1}(x_1,\psi_{-1}(x_2,x_3)) = f_{x_1 \otimes x_2 \otimes x_3}$	$\psi_0(x_1,x_2,x_3) = \psi_0(x_1,\psi_0(x_2,x_3)) = f_{x_1 \oplus x_2 \oplus x_3}$
$\psi_{-1}(x_1,x_2,-x_3) = \psi_{-1}(x_1,\psi_0(x_2,-x_3)) = f_{x_1 \otimes x_2 \otimes \neg x_3}$	$\psi_1(x_1,x_2,-x_3) = \psi_0(x_1,\psi_1(x_2,-x_3)) = f_{x_1 \oplus x_2 \oplus \neg x_3}$
$\psi_0(x_1,-x_2,-x_3) = \psi_{-1}(x_1,\psi_1(-x_2,-x_3)) = f_{x_1 \otimes \neg x_2 \otimes \neg x_3}$	$\psi_2(x_1,-x_2,-x_3) = \psi_0(x_1,\psi_2(-x_2,-x_3)) = f_{x_1 \oplus \neg x_2 \oplus \neg x_3}$
$\psi_1(-x_1,-x_2,-x_3) = \psi_0(-x_1,\psi_1(-x_2,-x_3)) = f_{\neg x_1 \otimes \neg x_2 \otimes \neg x_3}$	$\psi_3(-x_1,-x_2,-x_3) = \psi_1(-x_1,\psi_2(-x_2,-x_3)) = f_{\neg x_1 \oplus \neg x_2 \oplus \neg x_3}$

Constant configurations like $\psi_b(x_1,x_2,x_3) = 0$, if $b < -2$, and $\psi_b(-x_1,-x_2,-x_3) = 1$, if $b > 3$, are also representable. However, there are examples of un-representable networks with three inputs like the configuration $\psi_0(-x_1,x_2,x_3)$. Naturally, a neuron configuration–when representable–can by codified by different structures using an ŁNN. In particularly, we have the following:

Proposition 1. *If the neuron configuration* $\alpha = \psi_b(x_1,x_2,\ldots,x_{n-1},x_n)$ *is representable, but not constant, it can be codified in an ŁNN with the structure:*
$\alpha = \psi_{b_1}(x_1,\psi_{b_2}(x_2,\ldots,\psi_{b_{n-1}}(x_{n-1},x_n)\ldots))$*, where* $b_1,b_2,...,b_{n-1}$ *are integers, and* $b = b_1 + b_2 + ... + b_{n-1}$.

Since the n-nary operator ψ_b is commutative, variables $x_1,x_2,\ldots,x_{n-1},x_n$ could interchange its position in function $\alpha = \psi_b(x_1,x_2,\ldots,x_{n-1},x_n)$ without changing the operator output. By this we mean that, for a three input configuration when we permutate variables, we generate equivalent configurations: $\psi_b(x_1,x_2,x_3) = \psi_b(x_2,x_3,x_1) = \psi_b(x_3,x_2,x_1) = \ldots$. When these are representable, they can be codified in a string-based notation using logic connectives. But these different configurations generat equivalente formulas only if these formulas are disjunctive or conjunctive formulas, since these are the only formulas invariant to the variable, or its negation, position change. A disjunctive formula is a formula written using only disjunctions and negations. Similarly, a conjunctive formula is a formula written using only conjunctions or negations.

Proposition 2. *If* $\alpha = \psi_b(x_1,x_2,\ldots,x_{n-1},x_n)$ *is representable, it is the interpretation of a disjunctive formula or a conjunctive formula.*

This leaves us with the task of classifying a neuron configuration according to its symbolic representation. For that, we established a relationship using the configuration bias and the number of negative and positive weights.

Proposition 3. *[17] In a neuron configuration $\alpha = \psi_b(-x_1, -x_2, \ldots, -x_n, x_{n+1}, \ldots, x_m)$ with $m = n + p$ inputs and where n and p are, respectively, the number of negative and the number of positive weights, on the neuron configuration:*

1. *If $b = -p + 1$, the neuron is called a* conjunction *and it is an interpretation for formula $\neg x_1 \otimes \ldots \otimes \neg x_n \otimes x_{n+1} \otimes \ldots \otimes x_m$.*
2. *When $b = n$ the neuron is called a* disjunction*, and it is an interpretation for formula $\neg x_1 \oplus \ldots \oplus \neg x_n \oplus x_{n+1} \oplus \ldots \oplus x_m$.*

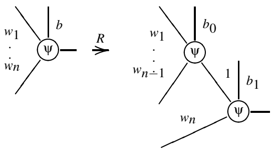
From the structure associated with this type of formula, we proposed the following structural characterization for representable neurons:

Proposition 4. *Every conjunctive or disjunctive $\alpha = \psi_b(x_1, x_2, \ldots, x_{n-1}, x_n)$, can be codified by an ŁNN*

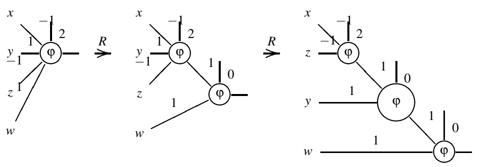
$$\beta = \psi_{b_1}(x_1, \psi_{b_2}(x_2, \ldots, \psi_{b_{n-1}}(x_{n-1}, x_n)\ldots)), \tag{1}$$

where $b = b_1 + b_2 + \cdots + b_{n-1}$ and $b_1 \leq b_2 \leq \cdots \leq b_{n-1}$.

This property can be translated in the following neuron rewriting rule,
linking



equivalent networks, when the values b_0 and b_1 satisfy $b = b_0 + b_1$ and $b_1 \leq b_0$, and are such that neither of the involved neurons has a constant output. This rewriting rule can be used to join equivalent configurations like the following:



Note that a representable CNN can be transformed by the application of rule R in a set of equivalent ŁNNs with the simplest neuron configuration. Then we have the following:

Proposition 5. *Un-representable neuron configurations are those transformed by rule R in at least two non-equivalent NNs.*

For instance, the un-representable configuration $\psi_0(-x_1, x_2, x_3)$, is transformed by rule R in three non-equivalent configurations: $\psi_0(x_3, \psi_0(-x_1, x_2)) = f_{x_3 \oplus (\neg x_1 \otimes x_2)}$, $\psi_{-1}(x_3, \psi_1(-x, x_2)) = f_{x_3 \otimes (\neg x_1 \otimes x_2)}$ and $\psi_0(-x_1, \psi_0(x_2, x_3)) = f_{\neg x_1 \otimes (x_2 \oplus x_3)}$. The representable configuration $\psi_2(-x_1, -x_2, x_3)$ is transformed by rule R on only two distinct but equivalent configurations: $\psi_0(x_3, \psi_2(-x_1, -x_2)) = f_{x_3 \oplus \neg (x_1 \otimes x_2)}$ and $\psi_1(-x_2, \psi_1(-x_1, x_3)) = f_{\neg x_2 \oplus (\neg x_1 \oplus x_3)}$.

For the extraction of knowledge from trained NNs, we translate neuron configurations in propositional connectives to form formulas. However, not all neuron configurations can be translated into formulas, but they can be approximated by them. To quantify the approximation quality, we defined the notion of interpretation λ-similar to a formula. Two neuron configurations, $\alpha = \psi_b(x_1, x_2, \dots, x_n)$ and $\beta = \psi_{b'}(y_1, y_2, \dots, y_n)$, are called λ-similar, in a $(m+1)$-valued ŁL, if

$$\lambda = e^{-\sum_{\bar{x} \in T} \frac{|\alpha(\bar{x}) - \beta(\bar{x})|}{\sharp T}}. \tag{2}$$

In this case we write $\alpha \sim_\lambda \beta$. If α is un-representable and β is representable, the second configuration is called *a representable approximation* to the first.

On the 2-valued ŁL (the Boolean logic case), we have for the un-representable configuration $\alpha = \psi_0(-x_1, x_2, x_3)$: $\psi_0(-x_1, x_2, x_3) \sim_{0.883} \psi_0(x_3, \psi_0(-x_1, x_2))$, $\psi_0(-x_1, x_2, x_3) \sim_{0.883} \psi_{-1}(x_3, \psi_1(-x_1, x_2))$, and $\psi_0(-x_1, x_2, x_3) \sim_{0.883} \psi_0(-x_1, \psi_0(x_2, x_3))$. In this case, the truth sub-tables of, formulas $\alpha_1 = x_3 \oplus (\neg x_1 \otimes x_2)$, $\alpha_1 = x_3 \otimes (\neg x_1 \otimes x_2)$ and $\alpha_1 = \neg x_1 \otimes (x_2 \oplus x_3)$ are both λ-similar to $\psi_0(-x_1, x_2, x_3)$, where $\lambda = 0.883$, since they differ in one position on 8 possible positions. The quality of these approximations was checked by analyzing the similarity level on other finite ŁLs. In every selected logic formula α_1, α_2 and α_3 had the same similarity level when compared to α: in the 3-valued logic, we have $\lambda = 0.8779$; in the 10-valued logic, $\lambda = 0.8798$; in 30-valued logic, $\lambda = 0.8814$; and in a 50-valued logic, $\lambda = 0.8818$. The level of similarity increases with the increase in the number of truth values.

For a more complex configuration like $\alpha = \psi_0(-x_1, x_2, -x_3, x_4, -x_5)$, we can derive the following configurations, using rule R: $\beta_1 = \psi_0(-x_5, \psi_0(x_4, \psi_0(-x_3, \psi_0(x_2, -x_1))))$, $\beta_2 = \psi_{-1}(x_4, \psi_{-1}(x_2, \psi_0(-x_5, \psi_0(-x_3, -x_1))))$, $\beta_3 = \psi_{-1}(x_4, \psi_0(-x_5, \psi_0(x_2, \psi_1(-x_3, -x_1))))$, $\beta_4 = \psi_{-1}(x_4, \psi_0(x_2, \psi_0(-x_5, \psi_1(-x_3, -x_1))))$. Since these configurations are not equivalents, we concluded that α is un-representable. In this case we can see a change in the similarity level between α and each β_i when the number of truth valued is changed: 2**-valued logic** $\alpha \sim_{0.8556} \beta_1$, $\alpha \sim_{0.9103} \beta_2$, $\alpha \sim_{0.5189} \beta_3$ and $\alpha \sim_{0.5880} \beta_4$; 5**-valued logic** $\alpha \sim_{0.8940} \beta_1$, $\alpha \sim_{0.9315} \beta_2$, $\alpha \sim_{0.4579} \beta_3$ and $\alpha \sim_{0.6326} \beta_4$; 10**-valued logic** $\alpha \sim_{0.9085} \beta_1$, $\alpha \sim_{0.9399} \beta_2$, $\alpha \sim_{0.4418} \beta_3$ and $\alpha \sim_{0.4991} \beta_4$. From observed similarity we selected β_2 as the best approximation to α. Note that its quality, as an approximation, improves when we increase the logics number of truth values.

In this sense, rule R can be used for configuration classification and configuration approximation. From an un-representable configuration, α, we can generate a finite set $S(\alpha)$, using rule R a, with representable networks similar to α. Given an $(n+1)$-valued logic, from that set of formulas we can select, as an approximation to α the formula having the interpretation more similar to α. This identification of un-representable configuration, with representable approximations, is used to transform networks with un-representable neurons into representable structures. The stress associated with this transformation characterizes the translation accuracy. Using the linearity of this type of function we have the following:

Proposition 6. *If β is the representable formula, generated by R, more similar to α in a $(n+1)$-valued ŁL then it is also the best approximation to α in an $(m+1)$-valued ŁL, when $m > n$.*

In practice we used 3-valued ŁL for the formula selection.

2.4 Neural Network Crystallization

Weights in CNNs assume the values -1 or 1. However, the usual learning algorithms process NNs weights, presupposing the continuity of the weights domain. Naturally, every NN with weights in $[-1,1]$ can be seen as an approximation to CNNs. The process of identifying an NN with weighs in $[-1,1]$ as a CNN is called *crystallization* and essentially consists of rounding each neural weight w_i to the nearest integer, less than or equal to w_i, denoted by $\lfloor w_i \rfloor$. In this sense the crystallization process can be seen as a pruning on the network structure, where links between neurons with weights near 0 are removed and weights near -1 or 1 are consolidated. However, this process is very "crispy", producing great losses of information. We need a smooth procedure to crystallize a network, in each learning iteration, to avoid the drastic reduction in learning performance. In each iteration we restricted the NN representation bias, making the network representation bias converge to a structure similar to a CNN. For that, we defined by *representation error* for a network N with weights $w_1,\ldots,w_n$, $\Delta(N) = \sum_{i=1}^{n}(w_i - \lfloor w_i \rfloor)$. When N is a CNN we have $\Delta(N) = 0$. Our *smooth crystallization* process results from the iteration of the following function:

$$\Upsilon_n(w) = sign(w).((\cos(1 - abs(w) - \lfloor abs(w) \rfloor).\frac{\pi}{2})^n + \lfloor abs(w) \rfloor), \qquad (3)$$

where $sign(w)$ is the sign of w and $abs(w)$ its absolute value. Denoting by $\Upsilon_n(N)$ the function has by input and output an NN, where the weights on the output network result in applying Υ_n to all the input network weights and neurons biases. Each interactive application of Υ_n produces a network progressively more similar to a CNN. Since, for every network N and $n > 0$, $\Delta(N) \geq \Delta(\Upsilon_n(N))$, we have the following:

Proposition 7. *Given a NNs N with weights in the interval* $[0,1]$. *For every* $n > 0$ *the function* $\Upsilon_n(N)$ *has, by fixed points, a CNNs.*

The convergence speed depends on parameter n. For our applications, we selected $n = 2$, based on the learning efficiency of a set of test formulas. Greater values for n impose stronger restrictions to learning.

3 Learning Propositions

We began the study of knowledge extraction using a CNN by reverse engineering a truth table. By this we mean that, for a given truth table on an $(n+1)$-valued ŁL, generated using a formula in the ŁL language, we will try to find its interpretation in the form of an ŁNN, and from it rediscover the original formula. For that we trained a feed-forward NN using a truth table. Our methodology trains progressively more complex networks until a crystallized network with good performance has been found. We use Algorithm 1 for the truth table reverse-engineering task.

Given part of a truth table we tried to find an ŁNN that codifies the data. For this we generated NNs with a fixed number of hidden layers (our implementation uses 3 hidden layers). When the process detects bad learning performances, it aborts the training, generating a new network with random heights. After a fixed number of tries, the network topology is changed. The number of tries for each topology depends on the number of network inputs. After trying to configure a set of networks for a given complexity with bad learning performance, the system tries to apply the selected back-propagation algorithm to a more complex set of networks. In the following we present a short description for the selected learning algorithm. If the continuous optimization process converges, i.e. if the system finds a network codifying the data, the network is crystallized. When the errors associated with this process increase, the system returns to the learning phase and tries to configure a new network. When the process converges and the resulting network can be codified as a crisp ŁNN, the system prunes the network. The goal of this phase is network simplification. For this, we selected the Optimal Brain Surgeon algorithm proposed by G. J. Wolf, B. Hassibi and D. G. Stork in [11].

Algorithm 1. Reverse-engineering Algorithm.

```
 1: Given a $(n+1)$-valued truth sub-table for a ŁL proposition.
 2: Define an inicial network complexity.
 3: Generate an inicial NN.
 4: Apply (the selected) Backpropagation algorithm using the data set.
 5: if the generated network has bad performance, then
 6:     If need, increase network complexity.
 7:     Try a new network. Go to 3.
 8: end if
 9: Do neural network crystallization using the crisp process.
10: if crystalized network performs badly, then
11:     Try a new network. Go to 3.
12: end if
13: Refine the crystalized network.
```

3.1 Training

Many efforts have been made to speed up the back-propagation algorithm. The Levenderg-Marquardt algorithm (LM) [18] ensued from the development of error back-propagation algorithm-dependent methods. It gives a good exchange between the speed of the Newton algorithm and the stability of the steepest descent method [19].

The basic LM algorithm adjusts the weights, on iteration $k+1$, using the update rule, $w_{k+1} = w_k - [J_k^T J_k + \mu.diag(J_k^T J_k)]^{-1} J_k^T e_k$, where J_k is the Jacobian matrix that contains first derivatives of the network errors e_k with respect to the weights w_k and μ is the learning rate. We changed the LM algorithm by applying a soft crystallization step after each LM update rule:

$$w_{k+1} = \Upsilon_2(w_k - [J_k^T J_k + \mu.diag(J_k^T J_k)]^{-1} J_k^T e_k). \tag{4}$$

This drastically improves the convergence to CNNs. In our methodology regularization is made using three different strategies: (1) using soft crystallization, where knowledge dissemination is restricted on the network, information is concentrated on some weights; (2) using crisp crystallization where only the heavier weights survive; (3) pruning the resulting crystallized network.

The regularization technique (3), avoids redundancies in the sense that the same or redundant information can be codified at different locations. We minimized this by selecting weights to eliminate. For this task, we used "Optimal Brain Surgeon"(OBS) method proposed by B. Hassibi, D. G. Stork, and G. J. Stork in [11], which uses the criterion of minimal increase in training error. It uses information from all second-order derivatives of the error function to perform network pruning. Our method is in no way optimal; it is just a heuristic, but it works well for learning CNNs.

4 Reverse-Engineering

Given an ŁNN it can be translated in the form of a string base formula if every neuron is representable. Proposition 3 defines a tool to translate from the connectionist representation to a symbolic representation. It is remarkable that, when the truth table sample used in the learning was generated by a formula, the Reverse Engineering algorithm converges to a representable ŁNN equivalent to the original formula, when evaluated on the cases used in the truth table sample.

When we generate a truth table in the 4-valued ŁL, using formula $(x_4 \otimes x_5 \Rightarrow x_6) \otimes (x_1 \otimes x_5 \Rightarrow x_2) \otimes (x_1 \otimes x_2 \Rightarrow x_3) \otimes (x_6 \Rightarrow x_4)$, it has 4096 cases, and the result of applying the algorithm is the 100% accurate NN A presented in figure 1. Using local interpretation, we may reconstruct the formula: $j_1 = \neg i_1 \otimes \neg i_2 \otimes \neg i_3 \otimes i_4 = \neg(\neg x_4 \otimes x_6) \otimes \neg(x_4 \otimes x_5 \otimes \neg x_6) \otimes \neg(x_1 \otimes x_2 \otimes \neg x_3) \otimes (\neg x_1 \oplus x_2 \oplus \neg x_5) = (x_4 \oplus \neg x_6) \otimes (\neg x_4 \oplus \neg x_5 \oplus x_6) \otimes (\neg x_1 \oplus \neg x_2 \oplus x_3) \otimes (\neg x_1 \oplus x_2 \oplus \neg x_5) = (x_6 \Rightarrow x_4) \otimes (x_4 \otimes x_5 \Rightarrow x_6) \otimes (x_1 \otimes x_2 \Rightarrow x_3) \otimes (x_1 \otimes x_5 \Rightarrow x2)$

Note, however, that the restriction imposed in our implementation of 3 hidden layers, wherein the last hidden layer has only one neuron, restricts the complexity of reconstructed formula. For instance, in order for $((x_4 \otimes x_5 \Rightarrow x_6) \oplus (x_1 \otimes x_5 \Rightarrow x_2)) \otimes (x_1 \otimes x_2 \Rightarrow x_3) \otimes (x_6 \Rightarrow x_4)$ to be codified in a 3-hidden-layer network, the last layer needs two neurons, one to codify the disjunction and the other to codify the conjunctions. When the algorithm was applied to the truth table generated in the 4-valued ŁL by using a stopping criterion of a mean square error less than 0.0007, it produced the representable NN B presented in figure 1. By this we may conclude that original formula can be approximated, or is λ-similar with $\lambda = 0.998$ to $j_1 = (x_6 \Rightarrow x_4) \otimes ((x_1 \otimes x_4 \otimes x_5) \Rightarrow (x_2 \oplus x_6)) \otimes (x_1 \otimes x_2 \Rightarrow x_3)$. Note that j_1 is 0.998-similar to the original formula in the 4-valued ŁL, but it is equivalent to the original in the 2-valued ŁL, i.e., in Boolean logic.

The fixed number of layers also imposes restrictions on the reconstruction of the formula. A truth table generated by $(((i_1 \otimes i_2) \oplus (i_2 \otimes i_3)) \otimes ((i_3 \otimes i_4) \oplus (i_4 \otimes i_5))) \oplus (i_5 \otimes i_6)$ requires at least 4 hidden layers, to be reconstructed; this is the number of levels required by the associated parsing tree.

Table below presents the mean CPU times needed to find a configuration with a mean square error of less than 0.002. The mean time is computed using 6 trials on a 5-valued truth ŁL for each formula. We implemented the algorithm using the MatLab NN package and executed it in an AMD Athlon 64 X2 Dual-Core Processor TK-53 at 1.70 GHz on a Windows Vista system with 959MB of memory. In this table the last two formula were approximated, since we restricted the structure for NNs to 3 hidden layers, for others each extraction process made equivalent reconstructions.

Formula	Mean	Stdev	Cases
$i_1 \otimes i_3 \Rightarrow i_6$	7.68	6.27	125
$i_4 \Rightarrow i_6 \otimes i_6 \Rightarrow i_2$	25.53	11.14	125
$((i_1 \Rightarrow i_4) \oplus (i_6 \Rightarrow i_2)) \otimes (i_6 \Rightarrow i_1)$	43.27	14.25	625
$(i_4 \otimes i_5 \Rightarrow i_6) \otimes (i_1 \otimes i_5 \Rightarrow i_2)$	51.67	483.85	3125
$((i_4 \otimes i_5 \Rightarrow i_6) \oplus (i_1 \otimes i_5 \Rightarrow i_2)) \otimes (i_1 \otimes i_3 \Rightarrow i_2)$	268.31	190.99	15625
$((i_4 \otimes i_5 \Rightarrow i_6) \oplus (i_1 \otimes i_5 \Rightarrow i_2)) \otimes (i_1 \otimes i_3 \Rightarrow i_2) \otimes (i_6 \Rightarrow i_4)$	410.47	235.52	15625

$$\begin{bmatrix} 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & -1 \\ 1 & 1 & -1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \\ -1 \\ 2 \end{bmatrix} \begin{array}{l} \neg x_4 \otimes x_6 \\ x_4 \otimes x_5 \otimes \neg x_6 \\ x_1 \otimes x_2 \otimes \neg x_3 \\ \neg x_1 \oplus x_2 \oplus \neg x_5 \end{array} \qquad \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & -1 \\ 1 & -1 & 0 & 1 & 1 & -1 \\ 1 & 1 & -1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ -1 \end{bmatrix} \begin{array}{l} x_4 \oplus \neg x_6 \\ x_1 \otimes \neg x_2 \otimes x_4 \otimes x_5 \otimes \neg x_6 \\ x_1 \otimes x_2 \otimes \neg x_3 \end{array}$$

$$\begin{bmatrix} -1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \end{bmatrix} \; \neg i_1 \otimes \neg i_2 \otimes \neg i_3 \otimes i_4 \qquad \begin{bmatrix} 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 0 \end{bmatrix} \; i_1 \otimes \neg i_2 \otimes \neg i_3$$

$$\begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} 0 \end{bmatrix} \; j_1 \qquad \begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} 0 \end{bmatrix} \; j_1$$

Fig. 1. NN A and NN B

5 Real Data

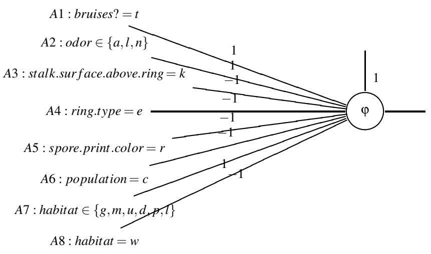
Extracting symbolic rules from a real data set can be a very different task than reverse-engineering the rule used on the generation of an artificial data set because, in the reverse engineering task, we know the existence of a perfect description. In particular, we know the appropriate logic language to describe it, and we have no noise. The process of symbolic extraction from the real data set is made by establishing a stopping criterion and having a language bias defined by the extraction methodology. The expressive power of this language characterizes the learning algorithm plasticity. Very expressive languages produce good fitness to data, but usually bad generalization; and the extracted sentences usually are difficult to understand by human experts.

The described extraction process, when applied to real data, expresses the information using CNNs. This naturally means that the process searches for simple and understandable models for the data, able to be codified directly or approximated using ŁL language. The process gives preference to the simplest models and subjects them to a strong pruning criteria. With this strategy we avoid overfitting. The process, however, can be prohibitive to train complex models that have a great number of links. To avoid this, rule extraction must be preceded by a phase of attribute selection.

5.1 Mushrooms

Mushroom is a data set available in the *UCI Machine Learning Repository*. This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. We will try to find a rule, for determining the edibility of a mushroom, using the data set as a truth table. The data set has 8124 instances, defined using 22 nominally valued attributes presented in the table below. It has missing attribute values, 2480, all for attribute #11. In the data set 4208 instances (51.8%) are classified as edible, and 3916 (48.2%) are classified as poisonous.

N.	Attribute	Values
0	classes	edible=e, poisonous=p
1	cap.shape	bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
2	cap.surface	fibrous=f, grooves=g, scaly=y, smooth=s
3	cap.color	brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
4	bruises?	bruises=t, no=f
5	odor	almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
6	gill.attachment	attached=a, descending=d, free=f, notched=n
7	gill.spacing	close=c, crowded=w, distant=d
8	gill.size	broad=b, narrow=n
9	gill.color	black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
10	stalk.shape	enlarging=e, tapering=t
11	stalk.root	bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
12	stalk.surface.above.ring	ibrous=f, scaly=y, silky=k, smooth=s
13	stalk.surface.below.ring	ibrous=f, scaly=y, silky=k, smooth=s
14	stalk.color.above.ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p,red=e,white=w,yellow=y
15	stalk.color.below.ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
16	veil.type	partial=p, universal=u
17	veil.color	brown=n, orange=o, white=w, yellow=y
18	ring.number	none=n, one=o, two=t
19	ring.type	cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
20	spore.print.color	black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
21	population	abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
22	habitat	grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d



$$\begin{bmatrix} 0 & 1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix} \begin{matrix} A2 \otimes \neg A5 \otimes A7 \\ A2 \otimes A4 \otimes \neg A7 \end{matrix}$$

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 \end{bmatrix} \quad i_1 \oplus i_2$$

Fig. 2. NN A and NN B

We used an unsupervised filter that converted all nominal attributes into binary numeric attributes. An attribute with k values was transformed into k binary attributes. This produced a data set containing 111 binary attributes. After the binarization we used the described method to select relevant attributes for mushroom classification by fixing a weak stopping criterion. As a result, the method produced a model with 100% accuracy, depending on 23 binary attributes defined by values of: *odor, gill.size, stalk.surface.above.ring, ring.type, and spore.print.color*.

We used the values assumed by these attributes to produce a new data set. After 3 tries we selected the NN A presented in figure 2. This model has an accuracy of 100%. From it, and since attribute values in A2 and A3, as well as the values in A7 and A8 are auto-exclusive, we used propositions A1, A2, A3, A4, A5, A6 and A7 to define a new data set. When we applied our "reverse engineering" algorithm to this data set, having as stopping criterion a mean square error (*mse*) less than 0.003, the method produced the NN B presented in figure 2. This model codifies the proposition, $(A2 \otimes \neg A5 \otimes A7) \oplus (A2 \otimes A4 \otimes \neg A7)$ and misses the classification of 48 cases. It has 99.41% accuracy and can be interpreted as the rule for editable mushrooms given by: "A mushroom is edible if its *odor*=almond.OR.anise.OR.none and *spore.print.color*= black.AND.*habitat*=NOT.waste or *ring.type*=evanescent.AND.*habitat*=NOT.waste."

A more precise model can be produced, by restricting the stopping criteria. However, doing so in general, produces more complex propositions that are more difficult to understand. For instance, with a stopping criterion $mse < 0.002$, the systems

$$\begin{array}{lll}
\begin{bmatrix} 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & -1 & -1 & 1 \end{bmatrix} & \begin{bmatrix} 1 \\ -1 \\ 0 \\ -1 \end{bmatrix} & \begin{array}{l} \neg A4 \oplus A7 \\ A1 \otimes A2 \otimes \neg A4 \\ A7 \\ A2 \otimes \neg A5 \otimes \neg A6 \otimes A7 \end{array} \\
\begin{bmatrix} -1 & 0 & 1 & 0 \\ 1 & -1 & 0 & -1 \end{bmatrix} & \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \begin{array}{l} \neg i_1 \oplus i_3 \\ i_1 \otimes \neg i_2 \otimes \neg i_4 \end{array} \\
\begin{bmatrix} 1 & -1 \end{bmatrix} & \begin{bmatrix} 0 \end{bmatrix} & j_1 \otimes \neg j_2
\end{array}
\qquad
\begin{array}{lll}
\begin{bmatrix} -1 & 1 & -1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & -1 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} & \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} & \begin{array}{l} i_1 \text{ un-representable} \\ A4 \otimes A5 \otimes \neg A7 \\ i_3 \text{ un-representable} \end{array} \\
\begin{bmatrix} 1 & -1 & 1 \end{bmatrix} & \begin{bmatrix} 0 \end{bmatrix} & j_1 \text{ un-representable}
\end{array}$$

Fig. 3. NN A and NN B

generated the NN A presented in figure 3. It misses 32 cases, has an accuracy of 99.2%, and is easy to convert into a proposition. NN A can be used to interprete the following formula: $j_1 \otimes \neg j_2 = ((A4 \otimes \neg A7) \oplus A7) \otimes ((A4 \otimes \neg A7) \oplus (A1 \otimes A2 \otimes \neg A4) \oplus (A2 \otimes \neg A5 \otimes \neg A6 \otimes A7)))$.

Sometimes the algorithm converged to un-representable configurations like the NN B presented in figure 3, with 100% accuracy. The frequency of this type of configurations increases with the increase of required accuracy.

Using rule R and selecting the best approximation to each un-representable formula evaluated in the data set, we have: $i_1 \sim_{0.9297} ((\neg A1 \otimes A4) \oplus A2) \otimes \neg A3 \otimes \neg A6$, $i_3 \sim_{1.0} (A1 \oplus \neg A7) \otimes A2$ and $j_1 \sim_{0.9951} (i_1 \otimes \neg i_2) \oplus i_3$.

The extracted formula, $\alpha = (((((\neg A1 \otimes A4) \oplus A2) \otimes \neg A3 \otimes \neg A6) \otimes \neg(A4 \otimes A5 \otimes \neg A7)) \oplus ((A1 \oplus \neg A7) \otimes A2)$ is λ-similar, with $\lambda = 0.9951$ to the original NN. Formula α misses the classification for 40 cases. Note that the symbolic model is stable, and the bad performance of the i_1 representation does not affect the model.

6 Conclusions and Future Work

This methodology of codifying and extracting symbolic knowledge from an NN is very simple and efficient for the extraction of comprehensible rules from small data sets. It is, moreover, very sensible to attribute relevance. From a theoretical point of view, it is particularly interesting that restricting the values assumed by neuron weights restricts the information propagation in the network, thus allowing the emergence of patterns in the neuronal network structure. For the case of linear neuronal networks, having by activation function the identity truncate to 0 and 1, these structures are characterized by the occurrence of patterns in the neuron configuration, directly presentable as formulas in ŁL. Generated fuzzy rules might do a good approximation of the data, but often are not interpretable. In our point of view, the interpretability of such symbolic rules is strictly related to the type of fuzzy logic associated to the problem. When we applied our method to the extraction of rules from truth tables, generated on Product logic or on Gödel logic, this rules were very difficult to interpret. For the extraction of knowledge from this type of fuzzy logics an extraction process that is governed by the appropriated logic must be developed.

We are applying this methodology in fuzzy regression tree generation and to active data mining. We used CNN for finding splitting formulas in the regression algorithm pruning phase and on the generation of database queries used for model revision.

References

1. Hitzler, P., Hölldobler, S., Seda, A.K.: Logic programs and connectionist networks. Journal of Applied Logic 2, 245–272 (2004)
2. d'Avila Garcez, A.S.: Advances in neural-symbolic learning systems: Modal and temporal reasoning. In: Hammer, B., Hitzler, P. (eds.) Perspectives of Neural-Symbolic Integration. SCI, vol. 77. Springer, Heidelberg (2007)
3. Komendantskaya, E., Lane, M., Seda, A.K.: Connectionistic representation of multi-valued logic programs. In: Hammer, B., Hitzler, P. (eds.) Perspectives of Neural-Symbolic Integration. SCI, vol. 77. Springer, Heidelberg (2007)
4. Eklund, P., Klawonn, F.: Neural fuzzy logic programming. IEEE Translations on Neural Networks 3(5) (1992)
5. Fu, L.M.: Knowledge-based connectionism from revising domain theories. IEEE Trans. Syst. Man. Cybern. 23, 173–182 (1993)
6. Towell, G.G., Shavlik, J.W.: Knowledge-based artificial neural networks. Artif. Intell., 70, 119–165 (1994)
7. Gallant, S.I.: Neural Network Learning and Expert Systems. MIT Press, Cambridge (1994)
8. Towell, G.G., Shavlik, J.W.: Extracting refined rules from knowledge-based neural networks. Mach. Learn. 13, 71–101 (1993)
9. Castro, J., Trillas, E.: The logic of neural networks. Mathware and Soft Computing 5, 23–27 (1998)
10. Hagan, M.T., Menhaj, M.: Training feedforward networks with marquardt algorithm. IEEE Transaction on Neural Networks 5(6), 989–993 (1999)
11. Hassibi, B., Stork, D.G., Wolf, G.J.: Optimal brain surgeon and general network pruning. In: IEEE International Conference on Neural Network, vol. 4(5), pp. 740–747 (1993)
12. Jipsen, P.: An overview of generalised basic logic algebra. Neural Network World 13(5), 491–500 (2003)
13. Gerla, B.: Functional representation of many-valued logics based on continuous t-norms. PhD thesis, University of Milano (2000)
14. Hájek, P.: Fuzzy logic from the logical point of view. In: Bartosek, M., Staudek, J., Wiedermann, J. (eds.) SOFSEM 1995. LNCS, vol. 1012. Springer, Heidelberg (1995)
15. Amato, P., Nola, A.D., Gerla, B.: Neural networks and rational łukasiewicz logic. IEEE Transaction on Neural Networks 5(6), 506–510 (2002)
16. Mundici, D.: A constructive proof of macnaughton's theorem in intinite-valued logics. Journal of Symbolic Logic 59, 596–602 (1994)
17. Dubois, D., Prade, H.: Fundamentals of fuzzy sets. Kluwer, Dordrecht (2000)
18. Charalambous, C.: Conjugate gradient algorithm for efficient training of artificial neural networks. IEEE Proceedings 139(3), 301–310 (1992)
19. Battiti, R.: Frist- and second-order methods for learning between steepest descent and newton's method. Neural Computation 4(2), 141–166 (1992)

Wireless Signal and Information Tracking Using Fuzzy Logic

Eddie C.L. Chan, George Baciu, and S.C. Mak

Department of Computing, The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
{csclchan,csgeorge,csscmak}@comp.polyu.edu.hk
http://www.comp.polyu.edu.hk/lab/WINS

Abstract. Over the last decade, many commercial and government organizations as well as university campuses have deployed WLANs such as IEEE 802.11b. This has fostered a growing interest in location-based services and applications. Fuzzy logic can be applied to evaluate the behaviour of Wireless Local Area Networks (WLAN) received signal strength (RSS) and as well as to retrieve the location-aware information according to the preference of user. The behavior study of WLAN signal strength is a pivotal part of WLAN tracking analysis. Previous analytical model has not been addressed effectively for analyzing how the WLAN infrastructure affected the accuracy of tracking. In this paper, we first propose a novel fuzzy spatio-temporal topographic model. We applied the Nelder-Mead (NM) method to simplify our previous work on fuzzy color map into a topographic (line-based) map. Secondly, we propose a location-aware information retrieval application that travelers access the application with Apple's iPhones which also identify the user current location. We demonstrate our idea with 17,000 restaurants in Hong Kong and make use of fuzzy logic to return the favorable dinning place search result according to the user's preference. Our result shows that the new analytical model can provide a detail and quantitative strong representation of WLAN RSS.

Keywords: Wireless tracking, Topographic mapping, Fuzzy logic, Wi-Fi signal strength, iPhone.

1 Introduction

Wireless Local Area Networks (WLAN) tracking analysis is a crucial part for deploying the efficient indoor positioning system. The analytical models can be used to visualize the distribution of signal and help to improve the design of positioning systems, for example by eliminating installation of WLAN access points (APs) and shortening the sampling time of WLAN received signal strength (RSS) in location estimation. Recent research on WLAN RSS analytical model [1] and [2] are based on the accuracy of positioning systems and proximity graphs, such as Voronoi diagram, clustering graph. They assume the distribution of the RSS is in Gaussian and pair wise. Some research works [2], [3], and [4] ignores the radio signal properties. Such assumptions may ignore or distort the real behavior of RSS and provides inadequate and inaccurate RSS analysis.

The fuzzy visualization map concepts widely applied in other fields, such as temperature, rainfall and atmosphere. Topographic mapping has been also highly recognized as a comprehensive method to visualize geographical information, such as the reflectance of slope and terrain. NM method also is used in many other fields such as data mining [5] and antenna optimization [6]. Fuzzy, topographic and NM modelling could well be applied to modelling in WLAN RSS analytical model.

In this paper, we first propose a novel analytical model that provides a visualization of the RSS distribution. We make use of the Nelder-Mead (NM) method to simplify our pervious works on the multi-layer fuzzy color model [7] to topographic (line-based) model. We develop a topographic model as analytical tools for evaluating and visualizing where the RSS is denser and clustering different RSS in different topographic level.

Secondly, we propose a location-aware information retrieval system that travelers access the system with Apple's iPhones. We apply fuzzy logic to search for the dinning place according to the preference of the user. We also extend iPhone's positioning features into indoor environment with our previous work of Wi-Fi positioning [7],[8]. Our system could be used in the entire area of Hong Kong city (1,104 km^2) with more than 17,000 restaurants. The restaurant information includes types of food, price, name, latitude, longitude of the restaurant and ratings from users. The entire set of database is retrieved and updated from [9]. Figure 1 shows the interface of our location-aware Application using Apple's iPhone.

The proposed analytical model offers two benefits. First, it serves as a quicker reference and efficient analysis tool. Second, it can provide a detail and quantitative strong representation of WLAN RSS. The iPhone application offers four benefits. First, our iPhone application provides a hybrid, accurate and effective positioning across indoors and outdoors. Second, it provides more flexibility to the user and provide recommendations of the restaurant according to the distance, user's preferred price and food type. Third, the database is remotely connected from [9] which user can rate, access and update the restaurant information through internet. Finally, it is user-friendly with full mobility.

The rest of this paper is organized as follows: Section 2 presents the topographic model design. Section 3 describes the iPhone application implementation. Section 4 describes the fuzzy membership function of distance and price. Section 5 presents the experimental setup of large scale site RSS surveying in 9.34 hectare campus area over 2,000 access points. Section 6 discusses the analysis result of obstacles, human bodies and WLAN APs location. Finally, Section 7 offers our conclusion and future work.

2 Topographic Model Design

The basic idea of topographic model is to plot a curve connecting minimum points where the function has a same particular RSS value. The sets of APs are known as topographic line nodes. Topographic line nodes are the APs residing on the topographic lines around contour region. In this section, we introduce the major operations of topographic model including propagation-based algorithm, fuzzy membership function in our previous work [7], topographic line node measurement, Nelder-Mead method and topographic model generation.

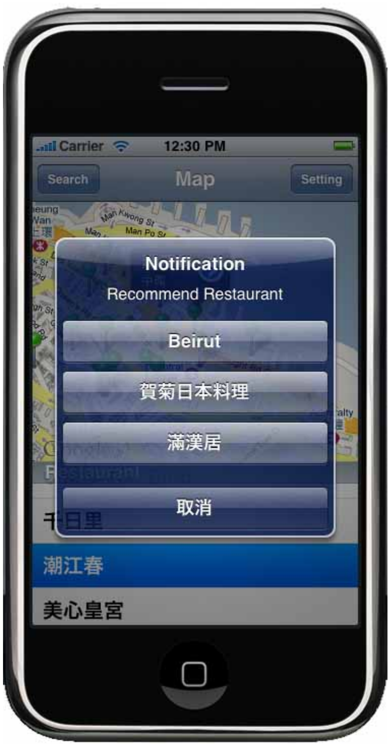


Fig. 1. Proposed Location-aware Application using Apple's iPhone

2.1 Propagation-Based Algorithm

The propagation-based algorithm [10] which is used to calculate the RSS as follows:

$$r_i(d_{i,k}) = r_0(d_0) - 10\alpha \log_{10}(d_{i,k}) - wallLoss \tag{1}$$

where $D = \{d_1...d_n | d_i \in \Re^n\}$ is a set of locations, $R = \{r_1...r_n | r_i \in \Re^n\}$ is a set of sampling LF vector respect to known d_i, α is the path loss exponent (clutter density factor) and $wallLoss$ is the sum of the losses introduced by each wall on the line segment drawn at Euclidean distance $d_{i,k}$.

2.2 Fuzzy Membership Function

In this subsection, our fuzzy membership function has been published in [7] and we will make use of it. Nonetheless, for completeness in the following we briefly describe.

Using fuzzy logic, the proposed model offers an enhanced LF hyperbolic solution that maps the RSS from a 0 to 1 fuzzy membership function. Instead of using a numeric value, the fuzzy logic determines the RSS as ”strong”, ”normal” and ”weak”.

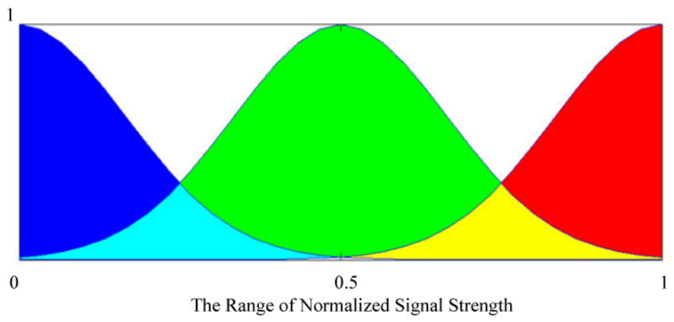


Fig. 2. Fuzzy membership graph for RSS

The normalization distribution is used to represent the fuzzy membership functions.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(\alpha-\mu)^2}{2\sigma^2}} \tag{2}$$

where $p(x)$ is the probability function, x is the normalized RSS, σ is the standard deviation of normalized signal normalized strength in a region, μ is the mean of signal strength in a region. The WLAN network is fully covered for the whole campus.

The membership function of term set, $\mu\,(\mathrm{RSSDensity}) = \{\mathrm{Red}, \mathrm{Green}, \mathrm{Blue}\}$. Red means the signal strength density is high, green means the signal strength is medium and blue means the signal strength density is low. The fuzzy set interval of blue is [0, 0.5], [0, 1] is green and [0.5, 1] is red.

For the blue region, we substitute $\sigma = 0.5$, $\mu = 0$.

$$\mu_{Blue}(0 < x < 0.5) = \frac{2}{\sqrt{2\pi}} e^{-2x^2} \tag{3}$$

For the green region, we substitute $\sigma = 0.5$, $\mu = 0.5$.

$$\mu_{Green}(0 < x < 1) = \frac{2}{\sqrt{2\pi}} e^{-2(x-\frac{1}{2})^2} \tag{4}$$

For the red region, we substitute $\sigma = 0.5$, $\mu = 1$.

$$\mu_{Red}(0.5 < x < 1) = \frac{2}{\sqrt{2\pi}} e^{-2(x-1)^2} \tag{5}$$

Figure 2 shows the fuzzy membership function. X-axis represents the normalized signal strength from 0 to 1 (from -93dBm to -15dBm). The width of membership function depends on the standard deviation of the RSS. The overlap area will be indicated by mixed colors. We can use different colored regions to represent the WLAN RSS distribution. Conceptually we define a spatio-temporal region as follows:

Assume that B is a finite set of RSS vector belonging to a particular color region, where $B = \{b_1...b_n | b_i \in \Re^n\}$, i.e., $b_i \in S$, $\forall S \in R$, and $\forall S \in [l, u]$, where l is the lower bound of fuzzy interval and u is a upper bound of fuzzy interval. To analyze the distribution surfaces S, there always exists a spatio-temporal mapping, $q : B \to S$.

$$q(x) = \int_S h(x)b(S)dS, \tag{6}$$

where $h(x)$ is the characteristic function of S, i.e.,

$$h(x) = \begin{cases} 1, x \in S \\ 0, x \notin S \end{cases} \tag{7}$$

and $b(S)$ is a weight function that specifies a prior on the distribution of surfaces S. We can explicitly define $b(S)$ by (1). By (3), (4), (5), (6), and (7), the RSS distribution can be illustrated.

2.3 Topographic Node

Each topographic node consists of three components and can be expressed as $< l, d, g >$, in which l represents topographic level, d represents the locations of Wi-Fi received signal, g represents the gradient direction of the RSS distribution. The spatial data value distribution mapped into the (x, y, l) space, where the co-ordinate (x, y) represents the location and $l = f(x, y)$ describes a function mapping from (x, y) co-ordinates to level l. The gradient vector g denotes the direction of RSS where to degrade in the space. The gradient vector can be calculated by:

$$g = -f'(x, y) = \left(\frac{\Delta f}{\Delta x}, \frac{\Delta f}{\Delta y}\right)^T \tag{8}$$

$$S = \frac{B - W}{2} \tag{9}$$

2.4 Nelder-Mead Method

The Nelder-Mead (NM) method is a commonly used nonlinear optimization algorithm for finding a local minimum of a function of several variables has been devised by Nelder and Mead [11]. It is a numerical method for minimizing an objective function in a many-dimensional space. Instead of using (1) and (2), we estimate the location by NM method.

First, we collect the location fingerprint, r with an unknown location (x, y). We define $f(n) = |n - r|$, where n is any location fingerprint. Second, we select three location fingerprints (LFs) to be three vertices of a triangle.

We initialize a triangle BGW and function f is to be minimized. Vertices B, G, and W, where $f(B)$ is the smallest value (best vertex), $f(G)$ is the medium value (good vertex), and $f(W)$ is a largest value (worst vertex). There are 4 cases when using NM method. They are reflection, expansion, contraction and shrink. We recursively use NM method until finding the point which is the local minimum (nearest) in B, G, W that they are the same value.

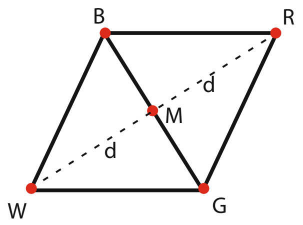


Fig. 3. Reflection Using the Point R

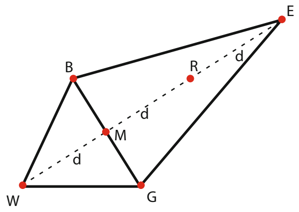


Fig. 4. Expansion Using the Point E

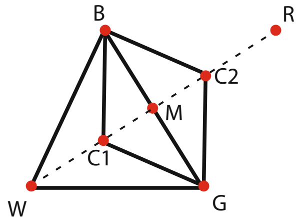


Fig. 5. Contraction Using the Point C

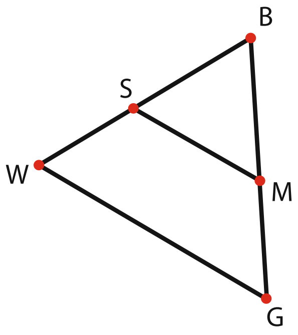


Fig. 6. Shrink toward B

The midpoint of the good side is

$$M = \frac{B+G}{2} \tag{10}$$

Reflection using the Point R. The function decreases as we move along the side of the triangle from W to B, and it decreases as we move along the side from W to G. Hence it is feasible that function f takes on smaller values at points that lie away from W on the opposite side of the line between B and G. We choose the point R that is obtained by "reflecting" the triangle through the side BG. To determine R, we first find the midpoint M of the side BG. Then draw the line segment from W to M and call its length d. This last segment is extended a distance d through M to locate the point R (See Figure 3). The vector formula for R is

$$R = M + (M - W) = 2M - W \tag{11}$$

Expansion Using the Point E. If the function value at R is smaller than the function value at W, then we have moved in the correct direction toward the minimum. Perhaps the minimum is just a bit farther than the point R. So we extend the line segment through M and R to the point E. This forms an expanded triangle BGE. The point E is found by moving an additional distance d along the line joining M and R (See Figure 4). If the function value at E is less than the function value at R, then we have found a better vertex than R. The vector formula for E is

Table 1. Nelder-Mead Method Procedure

```
IF f(R)<f(G), THEN Perform Case (i) {either reflect or extend}
ELSE Perform Case (ii) {either contract or shrink}

BEGIN {Case(i)}
IF f(B)<f(R) THEN
   replace W with R
ELSE
   compute E and f(E)
   IF f(E)<f(B) THEN
     replace W with E
   ELSE
     replace W with R
   ENDIF
ENDIF
END {Case (i)}

BEGIN {Case(ii).}
IF f(R)<f(W) THEN
   replace W with R
ENDIF

compute C = (W + M)/2
or C = (M + R)/2 and f (C)
IF f(C)<f(W) THEN
   replace W with C
ELSE
   compute S and f(S)
   replace G with M
ENDIF
END {Case (ii)}
```

$$E = R + (R - M) = 2R - M \tag{12}$$

Contraction using the Point C. If the function values at R and W are the same, another point must be tested. Perhaps the function is smaller at M, but we cannot replace W with M because we must have a triangle. Consider the two midpoints C_1 and C_2 of the line segments WM and MR, respectively (see Figure 5). The point with the smaller function value is called C, and the new triangle is BGC. Note. The choice between C_1 and C_2 might seem inappropriate for the two-dimensional case, but it is important in higher dimensions.

$$C_1 = \frac{M - W}{2} \tag{13}$$

$$C_2 = R - \frac{M - W}{2} \tag{14}$$

Shrink toward B. If the function value at C is not less than the value at W, the points G and W must be shrunk toward B (see Figure 6). The point G is replaced with M, and W is replaced with S, which is the midpoint of the line segment joining B with W.

2.5 Topographic Model Generation

We generate topographic model based on our previous work [7] and NM algorithm. We apply NM method to a many-dimensional RSS distribution space problem to simplify the fuzzy color map down to a contour (line-based) map.

First, we select three LFs to be three vertices of a triangle: $\boldsymbol{B}$, $\boldsymbol{G}$ and $\boldsymbol{W}$, where $\boldsymbol{B}$ is a location with high RSS (best vertex), $\boldsymbol{G}$ is a location with medium RSS

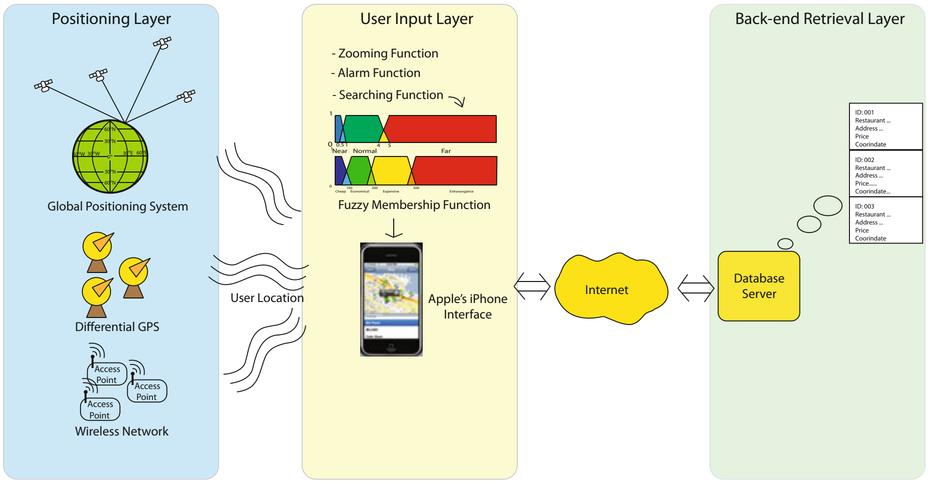


Fig. 7. System overview of location-aware information retrieval system using Apple's iPhone

(good vertex), and $\boldsymbol{W}$ is a location with the low RSS (worst vertex). The location vector of RSS at x_k,y_k use in function, $N(x, y)$. We use (1) to define $N(x, y)$. There are 4 cases when using NM method. They are reflection, expansion, contraction and shrink. We recursively use NM method until finding the point which is the local minimum in $\boldsymbol{B}$, $\boldsymbol{G}$ and $\boldsymbol{W}$ that they are the same value. Table 1 summarizes the procedure.

A contour function is then used to plot a curve connecting minimum points where the function has a same particular value. We normalize the minimum value between 0 and 1, and the contour line is 0.1 in each level.

3 iPhone Application Implementation

In this section, we introduce how our location-aware information application to be implemented. Figure 7 describes the system overview. There are three main layers in our application.

The first layer is the positioning layer. We use hybrid positioning technology to locate the user. iPhone 3GS has already included the features of DGPS. We further extend the positioning functions into indoors by Wi-Fi positioning techniques. In most of the cases, when the user stays in outdoor environment, DGPS would be used to estimate the user's location. When the iPhone cannot receive satellite signal or received satellite signal is very week, the proposed system will automatically turn to Wi-Fi positioning. The second layer is the user input layer which reads users' query, displays the restaurant information and identify the location in the Google map. We implement our system using iPhone as front-end user interface. iPhone is a touch device that it can easily control the zooming function of the map. We also implement alarm functions to remind and provide suitable choices of restaurants according to users' preference. The third layer is the back-end retrieval layer. This layer includes a location-aware database server

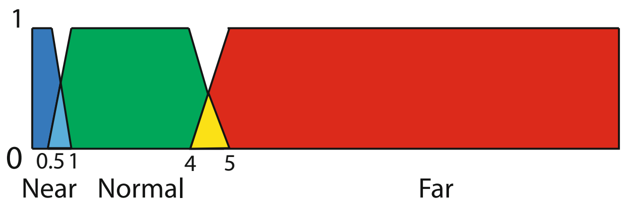


Fig. 8. Fuzzy membership graph for distance

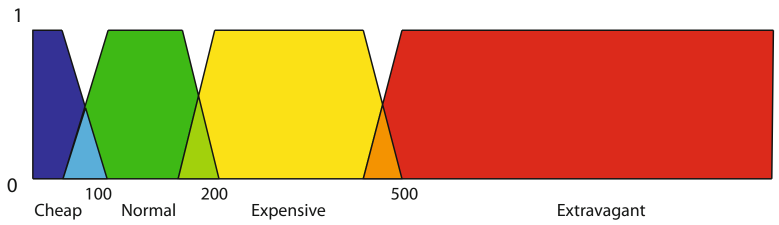


Fig. 9. Fuzzy membership graph for price

which stores the name, address, price, type and global co-ordinate of 17,000 restaurants. The database is updated and connected from [9].

4 Fuzzy Modeling for Distance and Price

Usually, human is sensible to some abstract concepts, such as far away, near, cheap or expensive. In this section, we make use of fuzzy logic to represent the distance and price. Fuzzy membership functions are used to represent the distance between the restaurant and the user location. The membership function of term set is $\mu(Distance) = \{Near, Normal, Far\}$. Figure 8 shows the fuzzy membership graph which X-axis represents the distance in kilometers and Y-axis represents the fuzzy membership from 0 to 1.

Similarly, fuzzy membership functions could be used to represent the price. $\mu(Price) = \{Cheap, Normal, Expensive, Extravagant\}$ is the membership function of term set. Figure 9 shows the fuzzy membership graph which X-axis represents the price in Hong Kong dollars and Y-axis represents the fuzzy membership from 0 to 1. The trapezoid function is used to represent the fuzzy membership functions of the distance and the price.

5 Experimental Setup

In this section, we describe experiment setup in 9.34 hectare campus area. We use the same setting as used in [12], [13], [1], [14], [10], [15] and [16]. RSS site survey measurement will be in The Hong Kong Polytechnic University (PolyU) campus. The approximate total area of the campus is 9.34 hectare. A standard laptop computer equipped

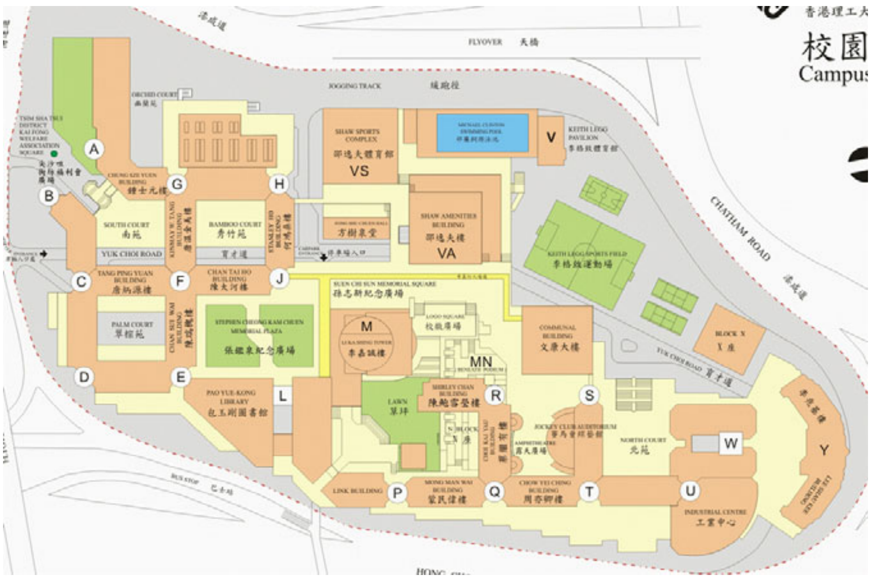


Fig. 10. The site plan for PolyU Campus with 27 buildings

with an Intel WLAN card and client manager software was used to measure samples of RSS from access points (APs) of PolyU campus.

There are basically 26 buildings from Core A to Core Z and 7 extra buildings with WLAN access. Each core building is covered by at least 13 APs. The received signal sensitivity of the WLAN card also limits the range of the RSS to be between -93 dBm and -15 dBm. Nevertheless, the highest typical value of the RSS is approximately -30 dBm at one meter from any AP. The sampling schedule is to collect the RSS data every 5 seconds. The vector of RSS data at each location forms the location fingerprint with around 20 RSS elements in the vector. Total 27 locations of measurement are chosen in the campus. Figure 10 show the 27 locations site plan. The radio channels used for each AP are channel 1, 6, and 11 respectively. The sampling will be taken with two periods of time, (7:30am-9:30am (leisure) and 4:30pm - 6:30pm (busy)). From [1], the presence or absence of people in the building significantly affects the RSS values. The duration of sampling was 2 weeks with total 12 days (from Mon to Sat). In mean while, temperature, weather, sampling time and humidity were recorded. The total number of RSS samples would be 12 days X 4 directions X 27 buildings X 20 APs X 2 times = 51,840.

6 Discussion and Analysis

In this section, we discuss the effect of the presence of human and LOS factor in our topographic model. There are three RSS features to be analyzed, LOS, the presence of human and RSS variation.

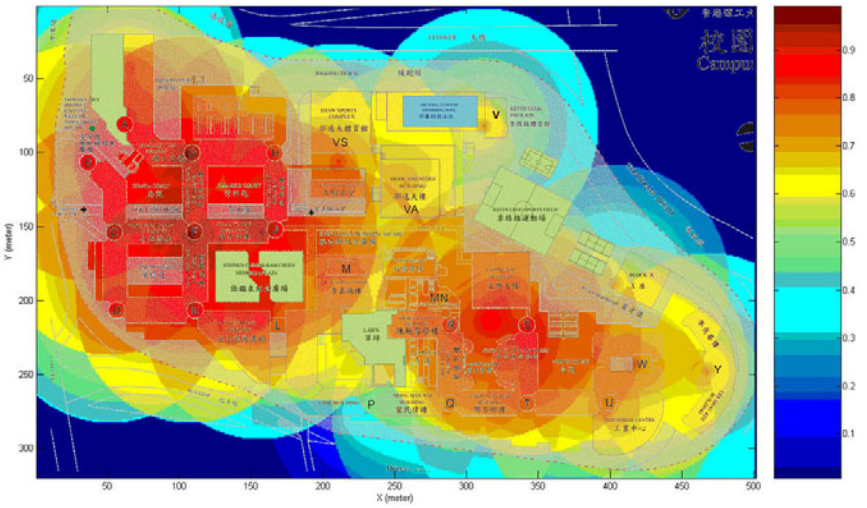


Fig. 11. Fuzzy RSS Distribution with the campus floor plan

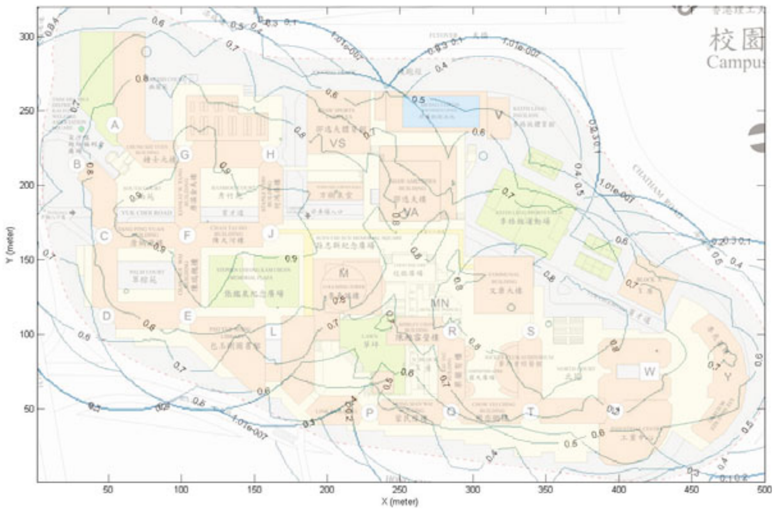
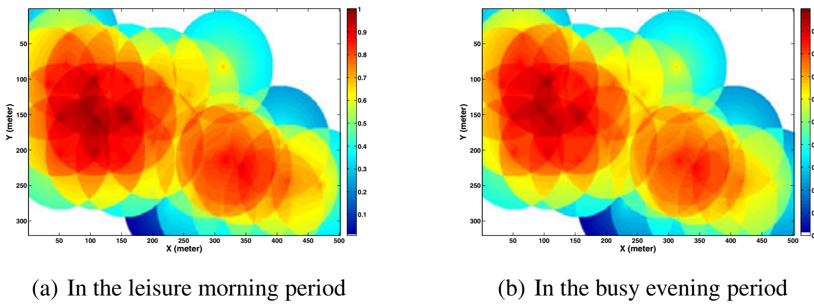


Fig. 12. Topographic RSS Distribution with the campus floor plan

6.1 Effect of LOS on RSS

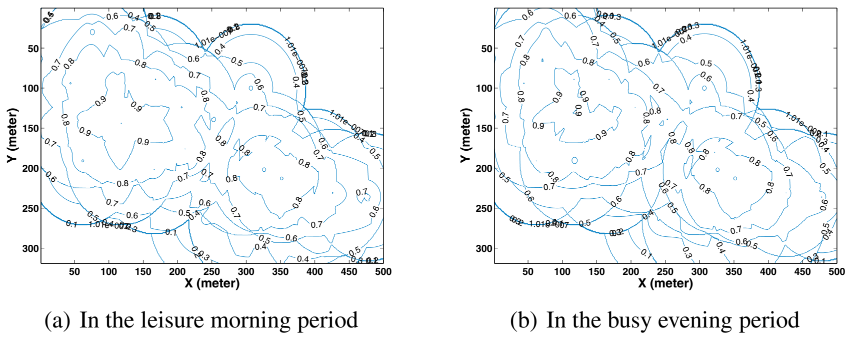
Figure 11 and 12 show the effect of LOS in two major clusters of RSS. The two major centers of high intensity locate at F core and S core.

The distance between F core to S core buildings is around 600m apart. The RSS should be covered evenly. Moreover, between M core (Lee Ka Shing Tower) to R core buildings (Shirley Chan Tower), the RSS distribution is relatively low. The heights of



(a) In the leisure morning period (b) In the busy evening period

Fig. 13. RSS Distribution in Fuzzy Analytical Model



(a) In the leisure morning period (b) In the busy evening period

Fig. 14. RSS Distribution in Topographic Model

two buildings in M core and R core are around 80m and 70m respectively. The distance between M to R core is around 200m apart.

As we can see the topographic map in Figure 14(a) and 14(b), the slope of contour line from M core to R core is steep in the edge area, it means that the RSS is weaken quickly in the middle from M core to R core due to NLOS effects. For LOS conditions, RSS should fit into log-normal distribution. A multi-story building in a campus area will experience lower signal strengths within tall buildings due to the absence of LOS propagation.

6.2 Behavior Study on the Human's Presence

As the previous section mention, we collected the RSS data in 2 periods, one is in the morning leisure period (7.30am-9.30am) and the other is in the busy evening period (4:30pm - 6.30pm). We would like to observe the difference between two periods. Figure 13 and 14 show the difference RSS pattern which the RSS collect in the two different time slot. We can see that there is significant change of the RSS value. Figure 14(a) shows the topographic region in 0.9 level is larger than Figure 14(b). We can observe the slope on Figure 14(b) degrades larger than Figure 14(a). As a result, it verifies the effect of the user's presence can affect the mean of the RSS value.

7 Conclusions

In this paper, we propose NM optimized topographic model for RSS distribution. The new model provides quicker references and efficient analysis tool for improving the design of WLAN infrastructure to achieve localization accuracy. In our university site experiment, we provide a spatial analytical model for WLAN tracking in a campus. The fuzzy topographic RSS analytical map provides easier understanding of WLAN RSS pattern in a region. The usage of model can improve the efficiency usage of WLAN infrastructure substantially.

In the future, wireless communications and mobility service provision will be characterized by global mobile access (terminal and personal mobility), a high quality of service with full coverage, and intelligible and simple access to multimedia services for voice and video via one user single terminal. In this paper, we propose a location-aware information retrieval system. The system helps the user to find suitable dinning place and provides accurate and robust positioning. We could further extend our applications into other domains, such as hotel reservation, movie booking and shopping in fashion store.

References

1. Kaemarungsi, K., Krishnamurthy, P.: Modeling of indoor positioning systems based on location fingerprinting. In: INFOCOM. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 2 (2004)
2. Swangmuang, N., Krishnamurthy, P.: Location Fingerprint Analyses Toward Efficient Indoor Positioning. In: Sixth Annual IEEE International Conference on Pervasive Computing and Communications, pp. 101–109 (2008)
3. Kjaergaard, M.B., Munk, C.V.: Hyperbolic Location Fingerprinting- A Calibration-Free Solution for Handling Differences in Signal Strength. In: Sixth Annual IEEE International Conference on Pervasive Computing and Communications, pp. 110–116 (2008)
4. Fang, S., Lin, T., Lin, P.: Location Fingerprinting In A Decorrelated Space. IEEE Transactions on Knowledge and Data Engineering 20(5), 685–691 (2008)
5. Satapathy, S., Murthy, J., Reddy, P., Katari, V., Malireddi, S., Kollisetty, V.: An Efficient Hybrid Algorithm for Data Clustering Using Improved Genetic Algorithm and Nelder Mead Simplex Search. In: International Conference on Computational Intelligence and Multimedia Applications, vol. 1 (2007)
6. Kolundzija, B., Olcan, D.: Antenna optimization using combination of random and Nelder-Mead simplex algorithms. In: Antennas and Propagation Society International Symposium, vol. 1. IEEE, Los Alamitos (2003)
7. Chan, C., Baciu, G., Mak, S.: Wireless Tracking Analysis in Location Fingerprint. In: 4th IEEE Wireless and Mobile Computing, Networking and Communications, IEEE WiMOB, pp. 214–220 (2008)
8. Chan, C.L., Baciu, G., Mak, S.: Using Wi-Fi Signal Strength to Localize in Wireless Sensor Networks. In: IEEE International Conference on Communications and Mobile Computing, CMC, pp. 538–542 (2009)
9. http://www.openrice.com.hk
10. Kaemarungsi, K., Krishnamurthy, P.: Properties of indoor received signal strength for WLAN location fingerprinting. In: The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, MOBIQUITOUS, pp. 14–23 (2004)

11. Mathews, J., Fink, K.: Numerical Methods Using MATLAB. Simon & Schuster, New York (1998)
12. Taheri, A., Singh, A., Emmanuel, A.: Location fingerprinting on infrastructure 802.11 wireless local area networks (WLANs) using Locus. In: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, pp. 676–683 (2004)
13. Kwon, J., Dundar, B., Varaiya, P.: Hybrid algorithm for indoor positioning using wireless LAN. In: IEEE 60th Vehicular Technology Conference on VTC2004-Fall, vol. 7 (2004)
14. Jan, R., Lee, Y.: An indoor geolocation system for wireless LANs. In: Proceedings of International Conference on Parallel Processing Workshops, pp. 29–34 (2003)
15. Wong, W., Ng, J., Yeung, W.: Wireless LAN positioning with mobile devices in a library environment. In: 25th IEEE International Conference on Distributed Computing Systems Workshops, pp. 633–636 (2005)
16. Bahl, P., Padmanabhan, V., Balachandran, A.: A Software System for Locating Mobile Users: Design, Evaluation, and Lessons. Online document, Microsoft Research (February 2000)

Redefinition of Mutual Information in the Fuzzy Sets Framework for Computational Genomics

Silvana Badaloni, Marco Falda, Paolo Massignan, and Francesco Sambo

Dept. of Information Engineering, University of Padova
Via Gradenigo 6/A - 35131 Padova, Italy
{name.surname}@unipd.it, massignan@tele2.it

Abstract. Mutual Information is a measure of correlation between two discrete random variables: the aim of this work is to provide a new definition of Mutual Information using concepts from Fuzzy Sets theory, to extend it to continuous variables. With this approach, we extended the model on which the well known REVEAL algorithm for Reverse Engineering of gene regulatory networks is based and we designed a new flexible version of it, called FuzzyReveal, able to avoid the loss of information caused by the binarization of the continuous biological variables.The predictive power of our new version of the algorithm is promising, being both significantly higher than the one of REVEAL and comparable with a state-of-the-art algorithm on a set of simulated problems.

1 Introduction

One of the main goals of studies on Genomics is to understand the mechanism of genetic regulation, which can be modelled as a gene regulatory network, a graph in which nodes represent genes or proteins and two or more nodes are connected if a regulatory relation exists between them. A widely used approach for inferring regulatory relations is based on the analysis of the Shannon Entropy and on the Mutual information of gene expression signals. This mechanism constitutes the basis of REVEAL [1], a well-known Reverse Engineering algorithm. This approach exploits a boolean model to represent gene regulatory networks in which each gene is modelled with a boolean variable True/False; its main aim is to gather boolean relations between time series of quantized gene expression values. However, the Boolean model on which the classical REVEAL algorithm is based is limited: a large amount of information is lost, when a real signal is approximated with just the two symbols 0 and 1.

In order to represent a real signal in a symbolic qualitative way, fuzzy methodologies can provide the basis for a more flexible model. In the present paper we will provide a new definition of Mutual Information in the fuzzy framework that will be used to extend in a fuzzy direction the REVEAL algorithm. In [2] the relationship between the notions of probability and fuzziness is deeply studied: in particular, an interpretation of fuzzy set theory in terms of conditional events and coherent conditional probabilities is proposed. We will apply this theory to re-define Mutual Information, which will be used in the core of the REVEAL algorithm: the modified algorithm will be called FuzzyReveal.

The paper is organized as follows: in Section 2 the concept of classical Mutual Information is recalled and the REVEAL algorithm described, in Section 3 first Conditional Probability is defined in terms of membership functions, then Mutual Information is rewritten in the new setting and the classical REVEAL algorithm updated accordingly. Section 4 reports an example of application.

2 Mutual Information and the REVEAL Algorithm

Given a discrete random variable x, taking values in the set X, its Shannon Entropy [3] is defined as

$$H(x) = -\sum_{\bar{x}\in X} p(\bar{x}) \log p(\bar{x}),$$

where $p(\bar{x})$ is the probability mass function $p(\bar{x}) = Pr(x = \bar{x})$, $\bar{x} \in X$. The joint entropy of a pair of variables x, y, taking values in the sets X, Y respectively, is

$$H(x,y) = -\sum_{\bar{x}\in X, \bar{y}\in Y} p(\bar{x},\bar{y}) \log p(\bar{x},\bar{y})$$

while the conditional entropy of x given y is defined as

$$H(x|y) = H(x,y) - H(x).$$

The Mutual Information of x, y is defined as $MI(x,y) = H(x) - H(x|y)$ and can be explicitly expressed as

$$MI(x,y) = \sum_{\bar{x}\in X, \bar{y}\in Y} p(\bar{x},\bar{y}) \log \frac{p(\bar{x},\bar{y})}{p(\bar{x})p(\bar{y})} \geq 0 \qquad (1)$$

When the two variables are independent, the joint probability distribution factorizes and the MI vanishes:

$$p(\bar{x},\bar{y}) = p(\bar{x})p(\bar{y}) \;\Rightarrow\; MI(x,y) = 0.$$

Mutual Information is therefore a measure of dependence between two discrete random variables and is used by the REVEAL algorithm [1] to infer causal relations between genes: for each gene in the genome, a time series of its rate of expression (called *gene profile*) is gathered from multiple DNA-microarray experiments; an example is depicted in Figure 1, with time samples on the x-axis and intensity of gene expression on the y-axis.

To apply REVEAL algorithm, gene profiles are then quantized in two levels, 0 (underexpressed) and 1 (overexpressed), and Mutual Information is computed between all possible pairs of genes. In the specific case probabilities are computed as the frequencies of the symbols 0 or 1 within a given sequence; since the sum of the probabilities being 0 or 1 must be equal to unity, $p(1) = 1 - p(0)$ and the formula for the entropy becomes:

$$H(x) = -p(0) \cdot \log(p(0)) - (1 - p(0)) \cdot \log(1 - p(0))$$

The joint probability is computed as a the probability of co-occurrence of two symbols.

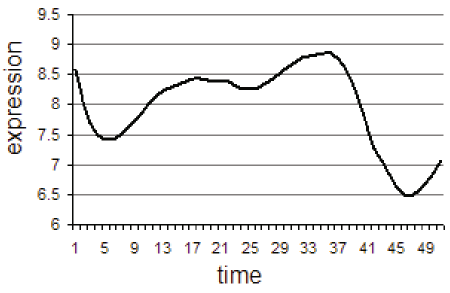


Fig. 1. Example of time series representing the expression of a gene

Example 1. Consider two random variables x and y, representing the quantization of two time series in two levels, 0 and 1; for each variable, consider two sequences of 10 symbols: $x' = \{0, 1, 1, 1, 1, 1, 1, 0, 0, 0\}$ and $y' = \{0, 0, 0, 1, 1, 0, 0, 1, 1, 1\}$. Then for variable x we obtain

$$p(0) = 0.4 \text{ and } p(1) = 0.6 = 1 - p(0)$$

that means 40 % of zeros and 60% of ones respectively. As for joint probabilities, in one case out of 10 $\exists i : x'_i = 0 \wedge y'_i = 0$, therefore

$$p(0, 0) = 0.1$$

The remaining combinations of symbols are $p(0, 1) = 0.3$, $p(1, 0) = 0.4$ and $p(1, 1) = 0.2$.

The algorithm infers causal relations between pairs whose MI is above a given threshold.

3 Fuzzy Extension of the REVEAL Algorithm

The classical REVEAL Algorithm is based on a Boolean model, therefore it has to approximate a real signal with just two symbols 0 and 1; it is clear that in this way much information is lost.

Using the Fuzzy Sets framework it is possible to obtain a more flexible and expressive model.

3.1 Membership Functions and Conditional Probability

In this paper the point of view of Coletti and Scozzafava [4,5] has been adopted.

Let x be a random quantity with range X, the family $\{x = \bar{x}, \bar{x} \in X\}$ is obviously a partition of the sample space Ω [6]; let then φ be any property related to the random quantity x: notice that a property, even if expressed by a proposition, does not single out an event, since the latter needs to be expressed by a non-ambiguous statement that can be either true or false. For this reason the event referred by a property will be indicated with E_φ, meaning “You claim E_φ” (in the sense of De Finetti [6]).

Coletti and Scozzafava state that a membership function can be defined as a Conditional Subjective Probability between two events E_φ and $x = \bar{x}$, meaning that "You believe that E_φ holds given $x = \bar{x}$".

$$\mu_{E_\varphi}(\bar{x}) = P(E_\varphi | x = \bar{x})$$

The membership degree $\mu_{E_\varphi}(\bar{x})$ is just the opinion of a real (or fictitious) person, for instance, a "randomly" chosen one, which is uncertain about it, whereas the truth-value of that event $x = \bar{x}$ is well determined in itself. Notice that conditional probability between events E_φ and $x = \bar{x}$ can be directly introduced rather than being defined as the ratio of the unconditional probabilities $P(E_\varphi \wedge \bar{x})$ and $P(x = \bar{x})$. From the same paper we report also the following example.

Example 2. Is Mary young? It is natural to think that You have some information about possible values of Mary's age, which allows You to refer to a suitable membership function of the fuzzy subset of "young people" (or, equivalently, of "young ages"). For example, for You the membership function may be put equal to 1 for values of the age less than 25, while it is put equal to 0 for values greater than 40; then it is taken as decreasing from 1 to 0 in the interval from 25 to 40. This choice of the membership function implies that, for You, women whose age is less than 25 are "young", while those with an age greater than 40 are not. The real problem is that You are uncertain on being or not "young" those women having an age between 25 and 40: the interest is in fact directed toward conditional events such as $E_{young} | x = \bar{x}$, with

$$E_{young} = \{\text{You claim that Mary is young}\}$$

$$\{x = \bar{x}\} = \{\text{the age of Mary is } \bar{x}\}$$

where $\bar{x}$ ranges over the interval [25, 40]. It follows that You may assign a subjective probability $P(E_{young} | x = \bar{x})$ equal to 0.2 without any need to assign a degree of belief of 0.8 to the event E_{young} under the assumption $x \neq \bar{x}$ (i.e., the age of Mary is not $\bar{x}$), since an additivity rule with respect to the conditioning events does not hold.

3.2 Fuzzy Mutual Information

The objective is to apply Coletti and Scozzafava theory to temporal gene profiles, therefore we introduce a set of properties Φ which describe qualitative aspects of the profiles, such as their "height" (*high*, *low*) or their "growth" (*increasing*, *decreasing*). Notice that the formula for Fuzzy Mutual Information that will be obtained is independent of the specific set chosen.

Exploiting the disintegration formula, the probability $\widetilde{P}$ of a single event for a property $\varphi \in \Phi$ can be written as

$$\begin{aligned}\widetilde{P}(E_{\varphi \in \Phi}) &= \sum_{\bar{x} \in X} P(E_\varphi | x = \bar{x}) \cdot P(x = \bar{x}) \\ &= \sum_{\bar{x} \in X} \mu_{E_\varphi}(\bar{x}) \cdot P(x = \bar{x})\end{aligned}$$

Since the Mutual Information relates two events (in our case relates two gene profiles) let, without loss of generalization, be $\Phi = \{\pi, \rho\}$. In the following we will write E_φ as $x = \varphi$.

The conjunctive probability for $x = \pi \wedge y = \rho$ is now required. According to [2,5] there is not an unique definition for the conditional probability $P(x = \pi \wedge y = \rho | x = \bar{x} \wedge y = \bar{y})$, called in the following p, for brevity. The probability p can assume any value such that

$$\max\{\mu_{E_\pi}(\bar{x}) + \mu_{E_\rho}(\bar{y}) - 1, 0\} \leq p \leq \min\{\mu_{E_\pi}(\bar{x}), \mu_{E_\rho}(\bar{y})\}$$

since it satisfies the coherence hypotheses [2]. Notice that the bounds for p are indeed T-Norms between the membership functions $\mu_{E_\pi}(\bar{x})$ and $\mu_{E_\rho}(\bar{y})$: p may in fact range between the Lukasievicz T-Norm and the minimum; in this work we show the results for the minimum, but good performance was achieved with many other values, such as product, Lukasievicz or the average between Lukasievicz and minimum. The probability that $x = \pi \wedge y = \rho$ can be defined, again in virtue of the disintegration property, as

$$\begin{aligned} &\widetilde{P}(x = \pi, y = \rho) = \\ &= \sum_{\bar{x} \in X} \sum_{\bar{y} \in Y} P(x = \pi \wedge y = \rho | x = \bar{x} \wedge y = \bar{y}) \\ &\quad \cdot P(x = \bar{x} \wedge y = \bar{y}) \\ &= \sum_{\bar{x} \in X} \sum_{\bar{y} \in Y} p \cdot P(x = \bar{x}, y = \bar{y}) \end{aligned}$$

The Fuzzy Mutual Information function can now be defined in a similar way w.r.t. the one defined in the Probabilistic setting (Formula 1) by replacing the probability distributions P with distributions $\widetilde{P}$ defined according to Coletti-Scozzafava's theory.

Definition 1. *Given two events x and y and a set of symbols Φ their Fuzzy Mutual Information is defined as*

$$\begin{aligned} &\widetilde{MI}(x, y) = \\ &\sum_{\varphi \in \Phi} \sum_{\varphi' \in \Phi} \widetilde{P}(x = \varphi, y = \varphi') \cdot \log \frac{\widetilde{P}(x = \varphi, y = \varphi')}{\widetilde{P}(x = \varphi) \cdot \widetilde{P}(y = \varphi')} \end{aligned}$$

This definition will be used to extend the classical REVEAL algorithm.

3.3 The Algorithm

The structure of the FuzzyReveal algorithm is similar to the classic REVEAL, but the extended definition of Fuzzy Mutual Information is used; its pseudo-code is reported in listing Algorithm 1.

Algorithm 1. Fuzzy Reveal

```
Input: $\mathcal{G} = \{x_1, \ldots, x_G\}$ a set of profile sequences, $\Phi$ the set of symbols, $N$ the
       number of pairs to return
Output: the first $N$ top-rated pairs
begin
    foreach $\bar{g}$ in $\mathcal{G}$ do
        foreach $\varphi$ in $\Phi$ do
            ▷ compute the membership function of the profile $x = \bar{g}$ w.r.t. the
            property $\varphi$
        end
    end
    $Rank \leftarrow \emptyset$
    foreach $x, y$ in $\mathcal{G}$ : $x \neq y$ do
        foreach $\pi, \rho$ in $\Phi$ do
            ▷ compute $\widetilde{P}(x = \pi, y = \rho)$
            ▷ compute $\widetilde{MI}(x, x)$ and $\widetilde{MI}(x, y)$
            ▷ compute $r_{xy} = \widetilde{MI}(x, y)/\widetilde{MI}(y, x)$
        end
        $Rank \leftarrow Rank \cup r_{ij}$
    end
    ▷ sort the pairs $\langle x, y \rangle$ according to $Rank$
    return the first N pairs
end
```

The parameter N can be set hypothesizing a scale-free topology for the underlying network: scale-free networks are sparse, with a number of edges that usually lies between V and $2V$, where V is the number of nodes [7].

4 Example of Application

The properties that have been considered to evaluate the proposed Fuzzy Mutual Information are:

- the value of the profile x at a given point $\bar{x}$ (high or low);
- the growth behavior of the profile x (increasing or decreasing).

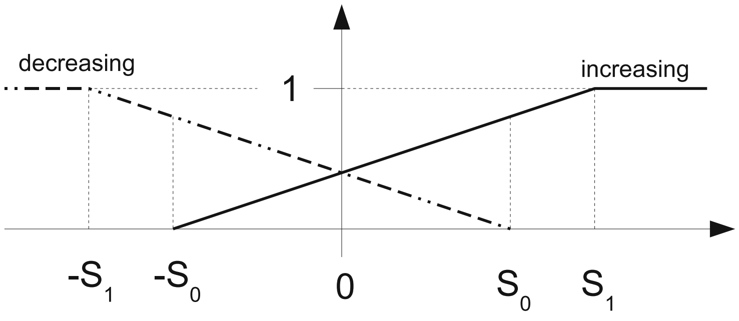
For each of these four events a membership function has been provided.

Definition 2. *the set Φ' is the set of qualitative aspects $\{"high", "low", "increasing", "decreasing"\}$.*

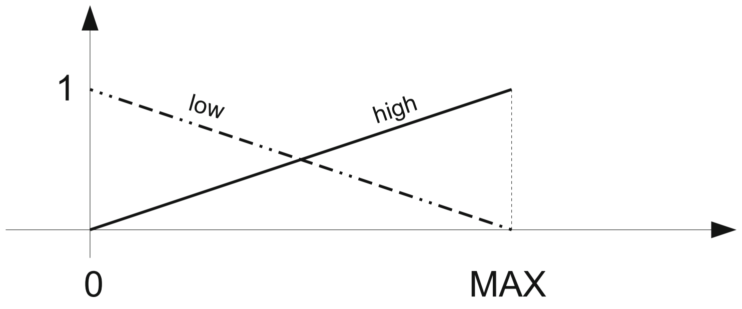
The membership functions for these qualitative aspects have been defined as

$$\mu_{high}(x) = \frac{x}{MAX}$$

$$\mu_{low}(x) = 1 - \mu_{high}(x)$$



(a) definitions for "increasing" and "decreasing".



(b) definitions for "high" and "low".

Fig. 2. Membership functions describing the growth behavior of a profile and its expression

$$\mu_{increasing}(x) = \begin{cases} 1 & \textit{if } x > S_1 \\ \frac{x+S_0}{S_1+S_0} & \textit{if } -S_0 \le x \le S_1 \\ 0 & \textit{otherwise} \end{cases}$$

$$\mu_{decreasing}(x) = \begin{cases} 1 & \textit{if } x < -S_1 \\ \frac{x-S_0}{S_1+S_0} & \textit{if } -S_1 \le x \le S_0 \\ 0 & \textit{otherwise} \end{cases}$$

where MAX is the maximum among all samples; S_1, S_2 are thresholds that shape the trapezoids (Figure 2), and they are applied to the angular coefficients of the series.

With this approach, a set of numerical values that represent the time series can be quantified using fuzzy levels, as in Figure 3.

Algorithm 1 has been evaluated using the Precision and Recall measures, defined as

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

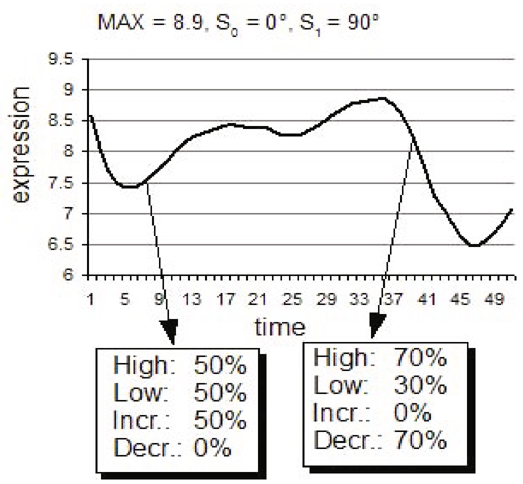


Fig. 3. Fuzzy values for two time series points

where TP represents the number of relations among genes that have been correctly identified by the algorithm (t*rue positives*), FP are the relations found by the algorithm but not representing real relations among genes (*false positives*), and finally FN are the real relations that the algorithm has not been able to find (*false negatives*).

To evaluate and compare the performance of different reverse-engineering approaches, transcriptional networks whose interactions are perfectly known should be used; since at present no biological network is known with sufficient precision to serve as a standard, quantitative assessment of reverse engineering algorithms can be accomplished using synthetic networks [8] or simulation studies [9,10].

To show the application of the new definition of Mutual Information we have generated two datasets for 12 genes and 50 time-points using an *ad hoc* simulator [11]. In Figure 4 the results are shown, together with the performances of a "state-of-the-art" algorithm (Aracne, [12]) and the classical version of REVEAL. It is possible to notice that there is a statistically significant improvement w.r.t. the classical algorithm[1]; this is mainly due to the fact that the classical REVEAL is based on Boolean networks, and so it uses just two values to represent gene expressions, while our approach allows describing time series intensity in a much better way and computing better similarity measures, since a whole range of values from 0 to 1 is used.

The comparison with Aracne is acceptable, since no statistically significant difference is observed in Precision and Recall.

5 Related Works

Soft computing tools, such as fuzzy sets, evolutionary strategies and neurocomputing, have been found to be helpful in providing low cost, acceptable solutions in the presence of various types of uncertainties when analysing gene regulatory networks data [13].

[1] Exact Wilcoxon two-sample tests, p-value < 0.05.

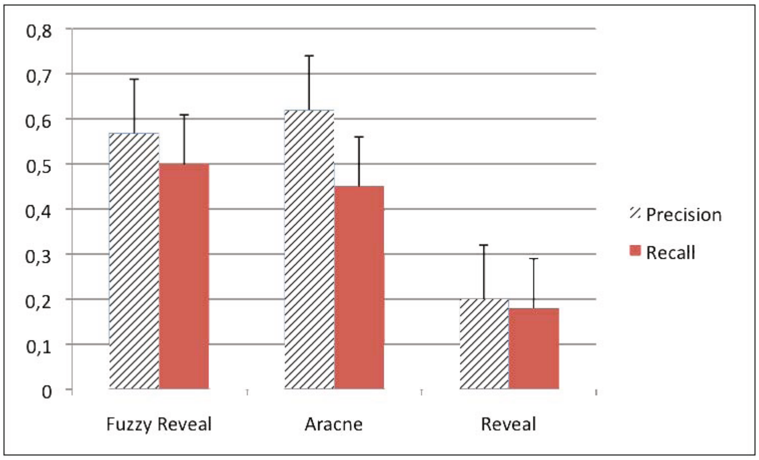


Fig. 4. Precision and recall measures of FuzzyReveal, Aracne and Classical Reveal algorithms

In [14], Mutual Information is computed between quantized profiles of gene expression, which are assigned to a fuzzy set of clusters: each profile can belong to different clusters, with different degrees of preference. More recently, Mutual Information extended using Fuzzy Sets and Rough Sets has been exploited to develop a new concept of equivalence partition matrix that allows for efficiently approximate the true marginal and joint distributions of continuous features from high dimensional microarray gene expression data [15].

Fuzzy rough sets and fuzzy Mutual Information have also been used in [16] for feature reduction, when selecting potential cancer genes from DNA-microarray experiments: given a subset of features, new features are added to the subset if their addition significantly increases the Mutual Information.

A Fuzzy Mutual Information measure is proposed in [17], where the authors follow the approach pioneered by De Luca and Termini; according to De Luca and Termini [18] an entropy function must satisfy four characteristic axioms, and in this way a Mutual Information function can be built. This is similar to what is done by Shannon using Probability theory instead of Fuzzy sets theory.

Other approaches to extend the REVEAL algorithm are present in the literature: [19] avoids the computation of Mutual Information, shifting the paradigm to the domain of *consistent pairs*, *i.e.* pairs $\langle regulators, regulated\ gene\rangle$ for which the same discrete value appears in the regulated gene every time the same combination of values appears in the set of regulators. In [20] this approach is further brought, defining a fitness function for putative causal relations, which allows the algorithm to rank pairs $\langle regulators, regulated\ gene\rangle$ in terms of distance from a consistent pair; the algorithm chooses, for each gene, the pair $\langle regulators, regulated\ gene\rangle$ closer to a consistent pair. The fitness function is specifically designed to tolerate quantization noise and variable regulatory delays.

6 Conclusions

In this paper we have considered the problem of Reverse Engineering and we have applied Coletti and Scozzafava results in order to replace expression profiles with qualitative descriptions. These descriptions are defined on a set of qualitative properties, and can assume different membership degrees w.r.t. a given property. Since the qualitative description comes from a random variable whose domain is finite, all classical results of Information Theory can be applied. We have extended the classical Mutual Information in a fuzzy direction and we have included it in the REVEAL algorithm thus obtaining the FuzzyReveal algorithm.

As for future directions, the application of this approach to real Genomics data will be the next step of the research. This will allow to have a better evaluation of performances with respect to both current biomedical experiments and noise typical of these data sets. A further possible improvement of this study could be the integration of a learning module for the automatic definition of the membership functions that describe the properties of the profiles.

References

1. Liang, S., Fuhrman, S., Somogyi, R.: Reveal: a general reverse engineering algorithm for inference of genetic network architectures. In: Pacific Symposium on Biocomputing, pp. 18–29 (1998)
2. Coletti, G., Scozzafava, R.: Conditional probability, fuzzy sets, and possibility: a unifying view. Fuzzy Sets and Systems 144, 227–249 (2004)
3. Shannon, C.E.: A mathematical theory of communication. The Bell Systems Technical Journal 27, 379–423 (1948)
4. Coletti, G., Scozzafava, R.: Probabilistic Logic in a Coherent Setting. Kluwer Academic Publishers, Dordrecht (2002)
5. Coletti, G., Scozzafava, R.: Conditional probability and fuzzy information. Computational Statistics & Data Analysis 51, 115–132 (2006)
6. de Finetti, B.: Teoria della Probabilità, Einaudi, Torino (1970)
7. Reka, A., Barabási, A.L.: Statistical mechanics of complex networks. Reviews of Modern Physics 74, 47–97 (2002)
8. Grilly, C., Stricker, J., Pang, W.L., et al.: A synthetic gene network for tuning protein degradation in saccharomyces cerevisiae. Mol. Syst. Biol. 3, 127 (2007)
9. Smith, V.A., Jarvis, E.D., Hartemink, A.J.: Evaluating functional network inference using simulations of complex biological systems. Bioinformatics 18, 216–224 (2002)
10. Mendes, P., Sha, W., Ye, K.: Artificial gene networks for objective comparison of analysis algorithms. Bioinformatics 19, 122–129 (2003)
11. Di Camillo, B., Toffolo, G., Cobelli, C.: A gene network simulator to assess reverse engineering algorithms. Ann. N Y Acad. Sci. 1158, 125–142 (2009)
12. Margolin, A.A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Dalla Favera, R., Califano, A.: Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. BMC Bioinformatics 7 (2006)
13. Mitra, S., Hayashi, T.: Bioinformatics with soft computing. IEEE Transactions on Systems, Man, and Cybernetics 36, 616–635 (2006)
14. Zhou, X., Wang, X., Dougherty, E.R., Russ, D., de Suh, E.: Gene clustering based on clusterwide mutual information. Journal of Computational Biology 11, 147–161 (2004)

15. Maji, P., Pal, K.S.: Fuzzy-rough sets for information measures and selection of relevant genes from microarray data. IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics (2010) (retrieved on April 2010) (to appear)
16. Xu, F., Miao, D., Wei, L.: Fuzzy-rough attribute reduction via mutual information with an application to cancer classification. Computers and Mathematics with Applications (2008)
17. Ding, S.F., Xia, S.X., Jin, F.X., Shi, Z.Z.: Novel fuzzy information proximity measures. Journal of Information Science 33, 678–685 (2007)
18. De Luca, A., Termini, S.: A definition of a non-probabilistic entropy in the setting of fuzzy sets theory. Information and Control 20, 301–312 (1972)
19. Nam, D., Seo, S., Kim, S.: An efficient top-down search algorithm for learning boolean networks of gene expression. Machine Learning 65, 229–245 (2006)
20. Sambo, F., Di Camillo, B., Falda, M., Toffolo, G., Badaloni, S.: CNET: an algorithm for the inference of gene regulatory interactions from gene expression time series. In: Proceedings of the 14th Workshop on Intelligent Data Analysis in Medicine and Pharmacology IDAMAP 2009, Verona, Italy, pp. 23–28 (2009)

Exact Membership Functions for the Fuzzy Weighted Average

Pim van den Broek[1] and Joost Noppen[2]

[1] Department of Computer Science, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands
pimvdb@ewi.utwente.nl
[2] Computing Department,University of Lancaster, Infolab21 Southdrive
Lancaster, LA1 4WA, U.K.
j.noppen@comp.lancs.ac.uk

Abstract. The problem of computing the fuzzy weighted average, where both attributes and weights are fuzzy numbers, is well studied in the literature. Generally, the approach is to apply Zadeh's extension principle to compute α-cuts of the fuzzy weighted average from the α-cuts of the attributes and weights for fixed values of $\alpha \in [0..1]$; this means that all values of the membership functions of the fuzzy weighted average are computed separately. In this paper, we generalise this approach in such a way that α is considered to be a parameter; this enables us to compute exact analytical membership functions for the fuzzy weighted average in case the attributes and weights are triangular or trapeizoidal fuzzy numbers. To illustrate the power of our algorithms, they are applied to the examples from the literature, providing exact membership functions in each case.

Keywords: Fuzzy weighted average, Membership functions.

1 Introduction

In multiple criteria decision making problems, values of decision variables are weighted averages of criteria ratings. Often the rating criteria and their corresponding importance weights are vague, and are therefore represented by fuzzy numbers. Then the values of the decision variables which are determined by them are fuzzy numbers as well; they are fuzzy weighted averages of the criteria ratings.

The standard approach to the calculation of fuzzy weighted averages [1-11] is to apply the extension principle to the following weighted average function:

$$\text{wa}(x_1, .., x_n, w_1, .., w_n) = \sum_{i=1}^{n}(x_i * w_i) \Big/ \sum_{i=1}^{n} w_i \tag{1}$$

Here $x_1,..,x_n$ are real numbers, called attributes, and $w_1,..,w_n$ are non–negative real numbers, called weights.

Let $A_1,..,A_n$ and $W_1,..,W_n$ be triangular or trapezoidal fuzzy numbers. Their α-cuts, denoted by $[A_1]^\alpha,.., [A_n]^\alpha, [W_1]^\alpha,.., [W_n]^\alpha$, are closed intervals; the elements of

$[W_1]^\alpha$,.., $[W_n]^\alpha$ are non-negative. The α-cut of the fuzzy weighted average of attributes A_1,..,A_n with weights W_1,..,W_n is given by the set

$$\{wa(x_1,..,x_n,w_1,..,w_n) \mid x_1 \in [A_1]^\alpha,..,x_n \in [A_n]^\alpha, w_1 \in [W_1]^\alpha,..,w_n \in [W_n]^\alpha\} \quad (2)$$

This set is a closed interval and is computed by computing its extremal values. Algorithms for computing these extremal values, for fixed value of α, are given in the literature. In this paper we will give an algorithm to solve this problem analytically, i.e. we will show how to compute analytical expressions for the extremal values of eq. (2). Also an algorithm for the computation of the inverses of these expressions is given; this enables us to calculate analytically the exact membership functions of the fuzzy weighted average.

In Section 2 we will present an algorithm for the computation of the extremal values of eq. (2) for a fixed value of α, which is based on previous algorithms. This algorithm has an intuitive geometrical interpretation, and therefore it can be easily understood. In Section 3 this algorithm is generalised to give analytical solutions, for triangular or trapeizoidal weights and attributes, and it is shown how these solutions can be reversed to give exact membership functions of the fuzzy weighted average. In Section 4, the power of the algorithm is demonstrated by applying it to examples from the literature, giving exact membership functions of the fuzzy weighted average in all cases.

2 Computation of α-Cuts of the Fuzzy Weighted Average

Let us start with the computation of the minimal value of the set of eq. (2). It is shown by Dong and Wong [3] and by Liou and Wang [10] that the minimum of eq. (2) is among the elements where x_i is equal to $\min[A_i]^\alpha$ and w_i equals either $\min[W_i]^\alpha$ or $\max[W_i]^\alpha$. So the problem can be reformulated as the problem of finding the minimal element of the finite set

$$\{\sum_{i=1}^{n} (\min[A_i]^\alpha * w_i) \Big/ \sum_{i=1}^{n} w_i \mid w_1 \in \{\min[W_1]^\alpha , \max[W_1]^\alpha\},..,w_n \in \{\min[W_n]^\alpha ,\max[W_n]^\alpha\}\} \quad (3)$$

Let Q be the set of all 2^n mappings from the set {1,2,..,n} to the set {+,–}. These mappings can be seen as states. A state is a partition of the attributes in attributes with maximal weight and attributes with minimal weight. The set of eq. (3) can be denoted as

$$\{\sum_{i=1}^{n} (\min[A_i]^\alpha * W_i^\alpha(q)) \Big/ \sum_{i=1}^{n} W_i^\alpha(q) \mid q \in Q\} \quad (4)$$

where $W_i^\alpha(q)$ stands for $\max[W_i]^\alpha$ if q(i) = + and for $\min[W_i]^\alpha$ if q(i) = –.

We define the mapping FWA from states to the real numbers by

$$FWA(q) = \sum_{i=1}^{n} (\min[A_i]^\alpha * W_i^\alpha(q)) \Big/ \sum_{i=1}^{n} W_i^\alpha(q) \ (5) \quad (5)$$

Now the problem has become the problem of finding the state q for which FWA(q) is minimal. The first step of the algorithm consists of sorting the set $\{\min[A_i]^\alpha\ |1<=i<=n\}$. From now on we assume this set is sorted. As a consequence, $\min[A_1]^\alpha <=$ FWA(q) $<= \min[A_n]^\alpha$ for all $q \in Q$. Now let us consider what happens when q changes such that, for some i with 1<=i<=n, q(i) changes from – into +. This change means that the weight of $\min[A_i]^\alpha$ increases, whereas the other weights remain the same. As a consequence, FWA(q) will change in such a way that it moves towards $\min[A_i]^\alpha$: the absolute value of FWA(q) – $\min[A_i]^\alpha$ decreases, but the sign of FWA(q) – $\min[A_i]^\alpha$ remains the same.

Let q_{min} be the state for which FWA takes its minimal value. Then $q_{min}(i) = +$ if $\min[A_i]^\alpha <$ FWA(q_{min}) and $q_{min}(i) = –$ if $\min[A_i]^\alpha >$ FWA(q_{min}). We know that $q_{min}(1) = +$ and $q_{min}(n) = –$. Our algorithm to obtain q_{min} is as follows. Let q_i be the element of Q defined by $q_i(j) = +$ for 1<=j<i and $q_i(j) = –$ for i<=j<=n. Then $q_{min} = q_i$ for some i with 2<=i<=n. To determine q_{min}, start with state q_2. Let us consider a situation with n=6, which is depicted as follows:

```
•------ •------- •------ ∇------- •------ •------- •
1+       2-         3-       Fwa       4-       5-        6-
```

Here the first line shows the real axis with $\min[A_i]^\alpha$ for i=1..6, and the fuzzy weighted average (Fwa) in the present state q_2. The second line shows the indices i, and whether the corresponding weights are maximal (+) or minimal (-) in the present state. We have to change the state in such a way that Fwa becomes as small as possible. Changing the state means changing the weights from maximal to minimal or vice versa. Changing weights 1,4,5, or 6 would increase Fwa. Changing weights 2 or 3 decreases Fwa. Suppose we change weight 3. The value of Fwa decreases, but remains greater than $\min[A_3]^\alpha$. Next we change weight 2. The value of Fwa is further decreased, and can become less than $\min[A_3]^\alpha$.In this case Fwa would decrease even further by restoring the original weight 3. This situation would have been avoided if we had first changed weight 2. If this causes Fwa to become less than $\min[A_3]^\alpha$, the minimum is reached in state q_3. Otherwise, weight 3 is changed as well, and the minimum is reached in state q_4. So it is important that the weight changes are performed from left to right. In Liou & Wang [10] that weight is changed first that causes Fwa to decrease most, which is incorrect.

Let us now return to the general case. The initial state is q_2. Compare FWA(q_2) with $\min[A_2]^\alpha$. If FWA(q_2) $<= \min[A_2]^\alpha$ then $q_{min} = q_2$, else continue with q_3. If FWA(q_3) $<= \min[A_3]^\alpha$ then $q_{min} = q_3$, else continue with q_4, et cetera. This iteration will terminate, since FWA(q_n) $<= \min[A_n]^\alpha$. So, the algorithm to compute the minimum of FWA(q) for $q \in Q$ is

```
sort { $\min[A_i]^\alpha$ | 1<=i<=n};
i = 2;
while (FWA($q_i$) > $\min[A_i]^\alpha$) i = i+1;
return FWA($q_i$);
```

The computation of the maximal value of the set of eq. (2) is quite similar. We now assume that the set $\{\max[A_i]^\alpha\ |1<=i<=n\}$ is sorted. Note that this ordering may be different from the one above. In the definition of FWA (eq. (5)) we replace $\min[A_i]^\alpha$

by $\max[A_i]^\alpha$. Let q_{max} be the value of q for which FWA(q) is maximal. Then $q_{max}(i) = -$ if $\max[A_i]^\alpha < FWA(q_{max})$ and $q_{max}(i) = +$ if $\max[A_i]^\alpha > FWA(q_{max})$. In particular, $q_{max}(1) = -$ and $q_{max}(n) = +$. Let q^i be the element of Q defined by $q^i(j) = -$ for $1<=j<=i$ and $q^i(j) = +$ for $i<j<=n$. Then $q_{max} = q^i$ for some i with $1<=i<=n-1$. To determine q_{max}, start with $q^{(n-1)}$. Compare $FWA(q^{(n-1)})$ with $\max[A_{(n-1)}]^\alpha$. If $FWA(q^{(n-1)}) >= \max[A_{(n-1)}]^\alpha$ then $q_{max} = q^{(n-1)}$, else continue with $q^{(n-2)}$, et cetera. So, the algorithm to compute the maximum of FWA(q) for $q \in Q$ is

```
sort { $\max[A_i]^\alpha$ |1<=i<=n};
i = n-1;
while (FWA($q^i$) < $\max[A_i]^\alpha$) i = i-1;
return FWA($q^i$);
```

Let us illustrate the algorithm with a small example, with n=6.

i	$[A_i]^\alpha$	$[W_i]^\alpha$	$FWA(q_i)$
1	[1,4]	[1,3]	
2	[2,6]	[1,3]	23/8
3	[3,7]	[1,3]	27/10
4	[4,5]	[1,3]	33/12
5	[5,8]	[1,3]	41/14
6	[6,9]	[1,3]	51/16

i	$[A_i]^\alpha$	$[W_i]^\alpha$	$FWA(q^i)$
1	[1,4]	[1,3]	109/16
2	[4,5]	[1,3]	99/14
3	[2,6]	[1,3]	87/12
4	[3,7]	[1,3]	73/10
5	[5,8]	[1,3]	57/8
6	[6,9]	[1,3]	

In the left-hand table the α-cuts $[A_i]^\alpha$ and $[W_i]^\alpha$ for some fixed α are shown in the second and the third column, sorted according to $\min[A_i]^\alpha$. The fourth column shows the values of $FWA(q_i)$. The minimum is seen to be 27/10, being the first value from above in the fourth column which is not greater than the corresponding value for $\min[A_i]^\alpha$ (in the second column). In the right-hand table the α-cuts $[A_i]^\alpha$ are sorted according to $\max[A_i]^\alpha$. The fourth column shows the values of $FWA(q^i)$. The maximum is seen to be 73/10, being the first value from below in the fourth column which is not less than the corresponding value for $\max[A_i]^\alpha$.

The extremal values obtained above could also have been found by taking the smallest resp. highest values in the fourth columns of the tables; this is indeed the approach of Chiao [2]. So apparently we did not gain anything, except some geometrical insight, as explained above. However, as it will turn out in the next section, for obtaining an analytical solution it is crucial not to compare values of the fourth column ($FWA(q_i)$ or $FWA(q^i)$) among each other, but instead compare elements of the fourth column with elements in the second column ($[A_i]^\alpha$).

The computational complexity of our algorithm is O(n*ln(n)), due to the first step, the sorting of the elements. The second phase, whose computational complexity is O(n), could be optimized by replacing the linear search by binary search, resulting in computational complexity O(ln(n)), as in Lee & Park [8]; the overall computational complexity would remain O(n*ln(n)), however. Guu [6] has given an algorithm with computational complexity O(n), in which the sorting of the elements is avoided. We have tried to keep our algorithm as simple as possible, in order to be able to generalize it to obtain an analytical solution.

3 Analytical Solution for the Fuzzy Weighted Average

In this section we will show that the algorithm given in the previous section can be generalized to obtain an analytical solution for the membership function of the fuzzy weighted average. There have been two previous attempts to obtain an analytical solution. Dong and Wong [3] obtained an analytical solution for two small examples. A general method was not given, however. Their approach was to consider the partial derivatives with respect to w_i of eq. (1) in order to obtain the extremal values of this equation. Kao and Liu [7] followed the same approach, and applied it to the same two examples, but also failed to provide a general solution. Our approach is different. We will generalize the algorithm of the previous section, by considering α to be a parameter which ranges over the interval [0,1], instead of being some fixed value. Then, taking the values of the fuzzy weights and fuzzy attributes to be triangular and trapezoidal fuzzy numbers, the extremal values of their α-cuts are linear functions of α.

A trapezoidal fuzzy number will be denoted as a 4–tuple (a,b,c,d) where a,b,c and d are real numbers with $a<=b<=c<=d$. The trapezoidal number (a,b,c,d) has membership function μ, given by

$$\begin{array}{ll} \mu(x) = 0, & \text{if } x<=a \\ \mu(x) = (x-a)/(b-a), & \text{if } a<x<b \\ \mu(x) = 1, & \text{if } b<=x<=c \\ \mu(x) = (d-x)/(d-c), & \text{if } c<x<d \\ \mu(x) = 0, & \text{if } x>=d \end{array} \qquad (6)$$

The restriction of μ to [a,b] and [c,d] will be referred to as the left-hand side resp. the right-hand side of the trapezoidal number. A triangular fuzzy number is a trapezoidal number of the form (a,b,b,c), and will be denoted by the 3–tuple (a,b,c).

Let the fuzzy weights be given by $W_i = (w_i,x_i,y_i,z_i)$ and the fuzzy attributes by $A_i = (a_i,b_i,c_i,d_i)$. Then we have, for $0<=\alpha<=1$:

$$\begin{array}{c} \min[A_i]^\alpha = a_i + \alpha\,(b_i - a_i) \\ \max[A_i]^\alpha = d_i - \alpha\,(d_i - c_i) \\ \min[W_i]^\alpha = w_i + \alpha\,(x_i - w_i) \\ \max[W_i]^\alpha = z_i - \alpha\,(z_i - y_i) \end{array} \qquad (7)$$

Our first aim is to find a function of α which is the minimum of the set of eq. (4) for all α in the interval [0,1]. The first step of the algorithm consists of sorting the set of left-hand sides of the attributes $\{\min[A_i]^\alpha \mid 1<=i<=n\}$. However, this sorting is the same for each α only if these left-hand sides do not intersect. So, we compute all the values of α for which two left-hand sides intersect. Note that coinciding left-hand sides present no problem; therefore they are considered not to intersect each other. Since each left-hand side is a linear function of α (eq. 6a), each pair of left-hand sides can intersect for at most one value of α with $0<=\alpha<=1$. So, there can be at most $n(n-1)/2$ such intersections. In practice, however, it turns out that there are only few intersections, if any at all. The values of α where the intersections occur partition the interval [0,1] in at most $n(n-1)/2+1$ subintervals.

On each of these subintervals the left-hand sides of the A_i do not intersect and the set $\{\min[A_i]^\alpha \mid 1<=i<=n\}$ can be sorted independent of α. We will consider each of these subintervals separately. So, in this step the problem has been reduced to the problem of finding the minimum of the set of eq. (4) for all α in some subinterval [min,max] of [0,1] where the ordering of the set $\{\min[A_i]^\alpha \mid 1<=i<=n\}$ is independent of α.

The next step of the algorithm is to compare $FWA(q_2)$ with $\min[A_2]^\alpha$. For values of α for which $FWA(q_2) <= \min[A_2]^\alpha$ the minimum is $FWA(q_2)$, for the other values of α the computation will continue with q_3. This is done by determining the values of α with $0<=\alpha<=1$ for which $FWA(q_2) = \min[A_2]^\alpha$. From the definition of FWA (eq. (5)) it follows that this equation can be written as:

$$\sum_{i=1}^{n} (\min[A_i]^\alpha * W_i^\alpha(q_2)) = \min[A_2]^\alpha * \sum_{i=1}^{n} W_i^\alpha(q_2) \tag{8}$$

From the eq. (7) we find that both sides of this equation are second order polynomials in α. Therefore, solving eq. (8) is trivial, and there are at most two solutions. Those solutions partition the interval [min,max] in at most three subintervals. On each of these subintervals the result of the comparison $FWA(q_2) <= \min[A_2]^\alpha$ is independent of α. On intervals where $FWA(q_2) <= \min[A_2]^\alpha$, the analytical solution is obtained, which is equal to $FWA(q_2)$. On intervals where $FWA(q_2) >= \min[A_2]^\alpha$, the computation continues in state q_3. Iteration of this process leads to the analytical solution of the minimum of the set of eq. (4) for all α in the interval [min,max]. This solution generally consists of separate solutions for a finite number of subintervals of [min,max]. Repeating this procedure for each of the intervals which were determined in the first step of the algorithm gives the analytical solution of the minimum of the set of eq. (4) for all α in the interval [0,1].

The algorithm can be summarized as follows:

```
Calculate the intersections of the left-hand sides of the
attributes;
Partition [0,1] into subintervals according to these
intersections;
For each subinterval [min,max]
   Adapt the numbering such that { $\min[A_i]^\alpha$ |1<=i<=n} is
sorted;
   Exact solution on [min,max] is Proc ([min,max], 2);
```

where `Proc` is defined by

```
Proc (interval, i) ==
   Partition the interval into subintervals according to
   solutions of $FWA(q_i) = \min[A_i]^\alpha$ ;
   On subintervals where $FWA(q_i) <= \min[A_i]^\alpha$ the exact
   solution is $FWA(q_i)$;
   On subintervals where $FWA(q_i) >= \min[A_i]^\alpha$ the exact
   solution is Proc (subinterval, i+1);
```

Note that the key element in this algorithm is the comparison of $FWA(q_i)$ with $\min[A_i]^\alpha$, which leads to a second order polynomial equation to be solved. An algorithm which compares values of $FWA(q_i)$ for different values of i, would have led to a third order polynomial equation, which is much more difficult to solve.

The algorithm to calculate a function of α which is the maximum of the set of eq. (4) for all α in the interval [0,1] is quite similar. It can be summarized as follows:

```
Calculate the intersections of the right-hand sides of the attributes;
Partition [0,1] into subintervals according to these intersections;
For each subinterval [min,max]
    Adapt the numbering such that { $\max[A_i]^\alpha$ |1<=i<=n} is sorted;
    Exact solution on [min,max] is Proc ([min,max], n-1);
```

where `Proc` is defined by

```
Proc (interval, i) ==
    Partition the interval into subintervals according to
    solutions of $FWA(q^i) = \min[A_i]^\alpha$ ;
    On subintervals where $FWA(q^i) <= \min[A_i]^\alpha$ the exact
    solution is $FWA(q^i)$;
    On subintervals where $FWA(q^i) >= \min[A_i]^\alpha$ the exact
    solution is Proc (subinterval, i-1);
```

The second aim in this section is to show that the exact solutions for the minimum and maximum values of the α-cuts of the fuzzy weighted average can be inverted to give the exact membership function of the fuzzy weighted average. First we will show how to invert the exact solution for the minimum values. In the preceding step of the algorithm, the interval [0,1] has been partioned in a finite number of subintervals, and on each subinterval [min,max] the exact solution is given by FWA(q) for some $q \in Q$. When the eqs. (7a-7b) are substituted in eq. (5), we find that FWA(q) is a function f of α which takes the form

$$f(\alpha) = (a\alpha^2+b\alpha+c)/(d\alpha+e) \tag{9}$$

where a,b,c,d,and e are real numbers. The inverse of f is the exact left-hand side of the membership function of the fuzzy weighted average on the interval [f(min), f(max)]; it can be computed by solving α from the equation x = f(α), which can be written as

$$a\alpha^2 + (b - dx)\alpha + c - ex = 0 \tag{10}$$

Let us first consider the case where $a \neq 0$. Here the solution of eq. (10) is given by

$$\mu(x)= (dx-b\pm\sqrt{(dx-b)^2 - 4a(c-ex)}\ )/(2a) \tag{11}$$

where the ambiguity in the sign can be solved with the conditions

$$\begin{aligned} \mu(f(\min)) &= \min \\ \mu(f(\max)) &= \max \end{aligned} \tag{12}$$

Next consider the case where a = 0 and $dc \neq eb$. Here the eq. (10) is solved by

$$\mu(x) = (ex - c)/(b - dx) \tag{13}$$

on the interval

$$[(b^*\min + c)/(d^*\min + e), \quad (b^*\max + c)/(d^*\max + e)] \tag{14}$$

Finally consider the case where a = 0 and dc = eb. In this case (bα + c)/(dα + e) is independent of α, so the inverse does not exist. Then the membership function is non–continuous in x = c/e. This occurs for instance when all attributes and weights are crisp numbers (i.e. of the form (a,a,a)), leading to a crisp weighted average, whose membership function is not continuous.

This shows that in each case the exact solution for the minimum values of the α-cuts of the fuzzy weighted average can be inverted, giving the exact solution the left-hand side of the membership function of the fuzzy weighted average. The computation of the exact solution of the right-hand side of the membership function is similar.

4 Examples

In this section we will apply our algorithms to derive exact membership functions to the examples of fuzzy weighted averages which have appeared in the literature. In each case, the fuzzy attributes and fuzzy weights are listed, as well as their α-cuts. This listing, and the assignment of indices from 1 to n, is such that the left-hand sides of the fuzzy attributes are properly ordered on the leftmost subinterval of [0,1]. In most of the cases this is the proper ordering on [0,1], both for the left-hand sides and for the right-hand sides of the fuzzy attributes. On subintervals where the ordering is different, this is clearly indicated. Renumbering fuzzy attributes and fuzzy weights has not been given explicitely, however, so the reader should be aware of the fact that the indices in eq. (5) are with respect to the proper ordering, and not necessarily with respect to the listed ordering.

We have avoided the rounding of real numbers as much as seems reasonable. So, quotients and square roots have not been evaluated to decimal form, except where expressions would otherwise become too unwieldy. To discriminate between decimal notations which are exact and those which are approximations, the latter are followed by an asterisk (*).

Example 1

For this example exact membership functions have been derived by Dong and Wong [3] and by Kao and Liu [7]. There are 2 attributes and 2 weights; all of these are triangular fuzzy numbers.

$A_1 = (0,1,2)$	$[A_1]^\alpha = [\alpha, 2-\alpha]$	$W_1 = (0,0.3,0.9)$	$[W_1]^\alpha = [0.3\alpha, 0.9-0.6\alpha]$
$A_2 = (2,3,4)$	$[A_2]^\alpha = [2+\alpha, 4-\alpha]$	$W_2 = (0.4,0.7,1)$	$[W_2]^\alpha = [0.4+0.3\alpha, 1-0.3\alpha]$

Since there are only two attributes, and there are no intersections, the calculation is trivial: the minimum of the α-cuts of the fuzzy weighted average is $FWA(q_2) = (-0.3\alpha^2 + 1.9\alpha + 0.8)/(-0.3\alpha + 1.3)$ and the maximum is $FWA(q^1) = -1.6\alpha + 4$.

This leads to the following membership function for the fuzzy weighted average:

$\mu(x) = 0$	if $x <= 8/13$ or $x >= 4$
$\mu(x) = x/2 + 19/6 - (5/3)\sqrt{0.09x^2 - 0.42x+4.57}$	if $8/13 <= x <= 12/5$
$\mu(x) = -5x/8 + 5/2$	if $12/5 <= x <= 4$

Example 2

For this example exact membership functions have been derived by Dong and Wong [3] and by Kao and Liu [7]. The example is also treated by Guh, Hon and Lee [4], by Guh, Hon, Wang and Lee [5] and by Liou and Wang [10]. There are 3 attributes and 3 weights; all of these are triangular fuzzy numbers.

$A_1 = (0,1,2)$	$[A_1]^\alpha = [\alpha, 2-\alpha]$	$W_1 = (0,0.3,0.9)$	$[W_1]^\alpha = [0.3\alpha, 0.9-0.6\alpha]$
$A_2 = (2,3,4)$	$[A_2]^\alpha = [2+\alpha, 4-\alpha]$	$W_2 = (0.4,0.7,1)$	$[W_2]^\alpha = [0.4+0.3\alpha, 1-0.3\alpha]$
$A_3 = (4,5,6)$	$[A_3]^\alpha = [4+\alpha, 6-\alpha]$	$W_3 = (0.6,0.8,1)$	$[W_2]^\alpha = [0.6+0.2\alpha, 1-0.2\alpha]$

First, we calculate the minimum of the α-cuts of the fuzzy weighted average. $FWA(q_2) = (-0.1\alpha^2 + 3.3\alpha + 3.2)/(-0.1\alpha + 1.9)$. $FWA(q_2) <= 2+\alpha$ for $0<=\alpha<=3/8$, so $FWA(q_2)$ is the minimum for $0<=\alpha<=3/8$. $FWA(q_2) >= 2+\alpha$ for $3/8<=\alpha<=1$. For $3/8<=\alpha<=1$ the minimum is $FWA(q_3) = (-0.7\alpha^2 + 2.7\alpha + 4.4)/(-0.7\alpha + 2.5)$.

Next we calculate the maximum of the α-cuts of the fuzzy weighted average. $FWA(q^2) = (-0.4\alpha^2 - 0.8\alpha + 7.6)/(0.4\alpha + 1.4)$. $FWA(q^2) >= 4-\alpha$ for $0<=\alpha<=1$, so $FWA(q^2)$ is the maximum for $0<=\alpha<=1$.

This leads to the following membership function for the fuzzy weighted average:

$\mu(x) = 0$	if $x <= 32/19$ or $x >= 38/7$
$\mu(x) = x/2 + 16.5 - 5\sqrt{0.01x^2 - 0.1x+12.17}$	if $32/19 <= x <= 19/8$
$\mu(x) = x/2 + 27/14 - (5/7)\sqrt{0.49x^2 - 3.22x+19.61}$	if $19/8 <= x <= 32/9$
$\mu(x) = -x/2 - 1 + (5/4)\sqrt{0.16x^2 - 1.6x+12.8}$	if $32/9 <= x <= 38/7$

Example 3

This example is treated by Guh, Hon, Wang and Lee [5]. There are 4 attributes and 4 weights; all of these are triangular fuzzy numbers.

$A_1 = (0,1,2)$	$[A_1]^\alpha = [\alpha,2-\alpha]$	$W_1 = (0,0.3,0.9)$	$[W_1]^\alpha = [0.3\alpha,0.9-0.6\alpha]$
$A_2 = (2,3,4)$	$[A_2]^\alpha = [2+\alpha, 4-\alpha]$	$W_2 = (0.4,0.7,1)$	$[W_2]^\alpha = [0.4+0.3\alpha, 1-0.3\alpha]$
$A_3 = (4,5,6)$	$[A_3]^\alpha = [4+\alpha, 6-\alpha]$	$W_3 = (0.6,0.8,1)$	$[W_3]^\alpha = [0.6+0.2\alpha, 1-0.2\alpha]$
$A_4 = (5,6,7)$	$[A_4]^\alpha = [5+\alpha, 7-\alpha]$	$W_4 = (0.5,0.8,1)$	$[W_4]^\alpha = [0.5+0.3\alpha, 1-0.2\alpha]$

First, we calculate the minimum of the α-cuts of the fuzzy weighted average. $FWA(q_2) = (0.2\alpha^2 + 5.3\alpha + 5.7)/(0.2\alpha + 2.4)$. $FWA(q_2) <= 2+\alpha$ for $0<=\alpha<=1$. $FWA(q_3) = (-0.4\alpha^2 + 4.7\alpha + 6.9)/(-0.4\alpha + 3)$. $FWA(q_3) >= 4+\alpha$ for $0<= \alpha<=1$, so $FWA(q_3)$ is the minimum for $0<= \alpha<=1$.

Next, we calculate the maximum of the α-cuts of the fuzzy weighted average. $FWA(q^3) = (-0.6\alpha^2 - 0.4\alpha + 12.2)/(0.6\alpha + 2)$. $FWA(q^3) >= 6-\alpha$ for $0<=\alpha<=0.1$, so $FWA(q^3)$ is the maximum for $0<=\alpha<=0.1$. $FWA(q^3) <= 6-\alpha$ for $0.1<=\alpha<=1$.

FWA(q^2) = $(-0.2\alpha^2 - 3.2\alpha + 14.6)/(0.2\alpha + 2.4)$. FWA($q^2$) >= $4-\alpha$ for $0.1<=\alpha<=1$, so FWA(q^2) is the maximum for $0.1<=\alpha<=1$.

This leads to the following membership function for the fuzzy weighted average:

$\mu(x) = 0$ if $x <= 23/10$ or $x >= 61/10$
$\mu(x) = x/2 + 47/8 - (5/4)\sqrt{0.16x^2 - 1.04x + 33.13}$ if $23/10 <= x <= 112/26$
$\mu(x) = - x/2 - 8 + (5/2)\sqrt{0.04x^2 - 0.64x + 21.92}$ if $112/26 <= x <= 59/10$
$\mu(x) = -x/2 - 1/3 + (5/6)\sqrt{0.36x^2 - 4.32x + 29.44}$ if $59/10 <= x <= 61/10$

Example 4

This example is treated by Lee and Park [8]. There are 5 attributes and 5 weights; all of these are triangular fuzzy numbers.

$A_1 = (1,2,3)$ $[A_1]^\alpha = [1+\alpha, 3-\alpha]$ $W_1 = (1,2,5)$ $[W_1]^\alpha = [1+\alpha, 5-3\alpha]$
$A_2 = (2,5,7)$ $[A_2]^\alpha = [2+3\alpha, 7-2\alpha]$ $W_2 = (2,2.5,3)$ $[W_2]^\alpha = [2+\alpha/2, 3-\alpha/2]$
$A_3 = (6,8,9)$ $[A_3]^\alpha = [6+2\alpha, 9-\alpha]$ $W_3 = (4,7,9)$ $[W_3]^\alpha = [4+3\alpha, 9-2\alpha]$
$A_4 = (7,9,10)$ $[A_4]^\alpha = [7+2\alpha, 10-\alpha]$ $W_4 = (3,4,7)$ $[W_4]^\alpha = [3+\alpha, 7-3\alpha]$
$A_5 = (10,11,12)$ $[A_5]^\alpha = [10+\alpha, 12-\alpha]$ $W_5 = (2,3,4)$ $[W_5]^\alpha = [2+\alpha, 4-\alpha]$

First, we calculate the minimum of the α-cuts of the fuzzy weighted average. FWA(q_2) = $(7.5\alpha^2 + 60\alpha + 74)/(2.5\alpha + 16)$. FWA($q_2$) >= $2+3\alpha$ for $0<=\alpha<=1$. FWA(q_3) = $(4.5\alpha^2 + 61\alpha + 76)/(1.5\alpha + 17)$. FWA($q_3$) <= $6+2\alpha$ for $0<=\alpha<=1$, so FWA(q_3) is the minimum for $0<=\alpha<=1$.

Next, we calculate the maximum of the α-cuts of the fuzzy weighted average. FWA(q^4) = $(-5\alpha^2 + 15.5\alpha + 131)/(4.5\alpha + 14)$. FWA($q^4$) <= $10-\alpha$ for $0<=\alpha<=1$. FWA(q^3) = $(-\alpha^2 - 28.5\alpha + 171)/(0.5\alpha + 18)$. FWA($q^3$) >= $9-\alpha$ for $0 <= \alpha <= 9\sqrt{3}-15$, so FWA($q^3$) is the maximum for $0<=\alpha <= 9\sqrt{3}-15$. FWA($q^3$) <= $9-\alpha$ for $9\sqrt{3}-15<=\alpha<=1$. FWA(q^2) = $(4\alpha^2 - 78.5\alpha + 216)/(-4.5\alpha + 23)$. FWA($q^2$) >= $7-2\alpha$ for $9\sqrt{3}-15<=\alpha<=1$, so FWA(q^2) is the maximum for $9\sqrt{3}-15 <= \alpha <= 1$.

These results are in accordance with the results by Lee and Park in [7] for α=0 and α=1. However, their claim that the fuzzy weighted average is a fuzzy triangular number is incorrect. Instead, the membership function of the fuzzy weighted average is calculated to be:

$\mu(x) = 0$ if $x <= 76/17$ or $x >= 19/2$
$\mu(x) = x/6 - 61/9 + (1/9)\sqrt{2.25x^2 + 123x + 2353}$ if $76/17 <= x <= 283/37$
$\mu(x) = -9x/16 + 157/16 -(1/8)\sqrt{20.25x^2 - 338.5x + 2706.25}$ if $283/37 <= x <= 24-9\sqrt{3}$
$\mu(x) = -x/4 - 57/4 + (1/2)\sqrt{0.25x^2 - 43.5x + 1496.25}$ if $24-9\sqrt{3} <= x <= 19/2$

Example 5

This example is treated by Kao and Liu [7]. There are 3 attributes and 3 weights; the weights are are triangular fuzzy numbers and the attributes are trapeizoidal fuzzy numbers.

$A_1 = (-2,1,2,3)$ $[A_1]^\alpha = [-2+3\alpha, 3-\alpha]$ $W_1 = (0,0.3,0.9)$ $[W_1]^\alpha = [0.3\alpha, 0.9-0.6\alpha]$
$A_2 = (1,2,3,5)$ $[A_2]^\alpha = [1+\alpha, 5-2\alpha]$ $W_2 = (0.4,0.7,1)$ $[W_2]^\alpha = [0.4+0.3\alpha, 1-0.3\alpha]$
$A_3 = (2,3,6,7)$ $[A_3]^\alpha = [2+\alpha, 7-\alpha]$ $W_3 = (0.6,0.8,1)$ $[W_3]^\alpha = [0.6+0.2\alpha, 1-0.2\alpha]$

First, we calculate the minimum of the α-cuts of the fuzzy weighted average. $FWA(q_2) = (-1.3\alpha^2 + 5.6\alpha - 0.2)/(-0.1\alpha + 1.9)$. $FWA(q_2) <= 1+\alpha$ for $0 <= \alpha <= (19 - \sqrt{109})/12$, so $FWA(q_2)$ is the minimum for $0 <= \alpha <= (19 - \sqrt{109})/12$. $FWA(q_2) >= 1+\alpha$ for $(19 - \sqrt{109})/12 <= \alpha <= 1$, so for $(19 - \sqrt{109})/12 < =\alpha < =1$ the minimum is $FWA(q_3) = (-1.9\alpha^2 + 5.6\alpha + 0.4)/(- 0.7\alpha + 2.5)$.

Next we calculate the maximum of the α-cuts of the fuzzy weighted average. $FWA(q^3) = (-0.7\alpha^2 - 0.8\alpha + 9)/(0.4\alpha + 1.4)$. $FWA(q^3) >= 5-2\alpha$ for 0<= <=1, so $FWA(q^3)$ is the maximum for $0<=\alpha<=1$.

This leads to the following membership function for the fuzzy weighted average:

$\mu(x) = 0$ if $x <= -2/19$ or $x >= 45/7$
$\mu(x) = x/26 + 28/13 - (5/13)\sqrt{0.01x^2 - 8.76x + 30.32}$ if $-2/19 <= x <= (31-\sqrt{109})/12$
$\mu(x) = 7x/38 + 28/19 - (5/19)\sqrt{0.49x^2 - 11.16x + 34.4}$ if $(31 - \sqrt{109})/12 <= x <= 41/18$
$\mu(x) = 1$ if $41/18 <= x <= 75/18$
$\mu(x) = -2x/7 - 4/7 + (5/7)\sqrt{0.16x^2 - 3.28x + 25.84}$ if $75/18 <= x <= 45/7$

Example 6

This example is treated by Chiao[2]. There are 8 attributes and weights; all of these are fuzzy triangular numbers.

$A_1 = (0,0.1,0.53)$ $[A_1]^\alpha = [0.1\alpha, 0.53-0.43\alpha]$ $W_1 = (0.55,0.66,0.67)$ $[W_1]^\alpha = [0.55+0.11\alpha, 0.67-0.01\alpha]$

$A_2 = (3.78,6.12,8.63)$ $[A_2]^\alpha = [3.78+2.34\alpha, 8.63-2.51\alpha]$ $W_2 = (0.52,0.80,0.85)$ $[W_2]^\alpha = [0.52+0.28\alpha, 0.85-0.05\alpha]$

$A_3 = (4.56,5.85,7.17)$ $[A_3]^\alpha = [4.56+1.29\alpha, 7.17-1.32\alpha]$ $W_3 = (0.62,0.75,0.8)$ $[W_3]^\alpha = [0.62+0.13\alpha, 0.8-0.05\alpha]$

$A_4 = (5.08,6.11,7.88)$ $[A_4]^\alpha = [5.08+1.03\alpha, 7.88-1.77\alpha]$ $W_4 = (0,0.07,0.1)$ $[W_4]^\alpha = [0.07\alpha, 0.1-0.03\alpha]$

$A_5 = (5.51,7.22,9.59)$ $[A_5]^\alpha = [5.51+1.71\alpha, 9.59-2.37\alpha]$ $W_5 = (0.09,0.38,0.46)$ $[W_5]^\alpha = [0.09+0.29\alpha, 0.46-0.08\alpha]$

$A_6 = (7.14,7.21,7.27)$ $[A_6]^\alpha = [7.14+0.07\alpha, 7.27-0.06\alpha]$ $W_6 = (0.44,0.63,0.91)$ $[W_6]^\alpha = [0.44+0.19\alpha, 0.91-0.28\alpha]$

$A_7 = (8.14,9.85,10)$ $[A_7]^\alpha = [8.14+1.71\alpha, 10-0.15\alpha]$ $W_7 = (0.59,0.61,0.85)$ $[W_7]^\alpha = [0.59+0.02\alpha, 0.85-0.24\alpha]$

$A_8 = (8.67,9.26,10)$ $[A_8]^\alpha = [8.67+0.59\alpha, 10-0.74\alpha]$ $W_8 = (0.03,0.18,0.29)$ $[W_8]^\alpha = [0.03+0.15\alpha, 0.29-0.11\alpha]$

First, we calculate the minimum of the α-cuts of the fuzzy weighted average. There are 4 intersections of left-hand sides: A_2 and A_3 intersect at 78/105, A_2 and A_4 intersect at 130/131, A_5 and A_6 intersect at 163/164 and A_7 and A_8 intersect at 53/112. This means that we have to deal with subintervals [0,53/112], [53/112,78/105], [78/105,130/131], [130/131,163/164] and [163/164,1] separately.

Subinterval [0, 53/112]: Ordering $[A_1,A_2,A_3,A_4,A_5,A_6,A_7,A_8]$.
$FWA(q_2) = (1.5259\alpha^2 + 9.7195\alpha + 13.493)/(1.12\alpha + 2.96)$.
$FWA(q_2) >= 3.78 + 2.34\alpha$ for $0 <= \alpha <= 53/112$.
$FWA(q_3) = (0.7537\alpha^2 + 9.2443\alpha + 14.7404)/(0.79\alpha + 3.29)$.
$FWA(q_3) <= 4.56+1.29\alpha$ for $0 <= \alpha <= 0.19463^*$.
$FWA(q_3)$ is the minimum for $0 <= \alpha<= 0.19463^*$.
$FWA(q_3) >= 4.56+1.29\alpha$ for $0.19463^* <= \alpha <= 53/112$.
$FWA(q_4) = (0.5215\alpha^2 + 8.6557\alpha + 15.5612)/(0.61\alpha + 3.47)$.
$FWA(q_4) <= 5.08 + 1.03\alpha$ for $0.19463^* <= \alpha <= 53/112$.
$FWA(q_4)$ is the minimum for $0.19463^* <= \alpha <= 53/112$.

Subinterval [53/112,78/105]: Ordering $[A_1,A_2,A_3,A_4,A_5,A_6,A_8,A_7]$.
$FWA(q_2) = (1.5259\alpha^2 + 9.7195\alpha + 13.493)/(1.12\alpha + 2.96)$.
$FWA(q_2) >= 3.78+2.34\alpha$ for $53/112<=\alpha<=78/105$.
$FWA(q_3) = (0.7537\alpha^2 + 9.2443\alpha + 14.7404)/(0.79\alpha + 3.29)$.
$FWA(q_3) >= 4.56+1.29\alpha$ for $53/112 <= \alpha <= 78/105$.
$FWA(q_4) = (0.5215\alpha^2 + 8.6557\alpha + 15.5612)/(0.61\alpha + 3.47)$.
$FWA(q_4) <= 5.08+1.03\alpha$ for $53/112 <= \alpha <= 78/105$.
$FWA(q_4)$ is the minimum for $53/112 <= \alpha <= 78/105$.

Subinterval [78/105, 130/131]: Ordering $[A_1,A_3,A_2,A_4,A_5,A_6,A_8,A_7]$.
$FWA(q_2) = (1.5259\alpha^2 + 9.7195\alpha + 13.493)/(1.12\alpha + 2.96)$.
$FWA(q_2) >= 4.56 + 1.29\alpha$ for $78/105 <= \alpha <= 130/131$.
$FWA(q_3) = (1.2937\alpha^2 + 9.1309\alpha + 14.3138)/(0.94\alpha + 3.14)$.
$FWA(q_3) <= 3.78+2.34\alpha$ for $0.93436^* <= \alpha <= 130/131$.
$FWA(q_3)$ is the minimum for $0.93436^* <= \alpha <= 130/131$.
$FWA(q_3) >= 3.78+2.34\alpha$ for $78/105 <= \alpha <= 0.93436^*$.
$FWA(q_4) = (0.5215\alpha^2 + 8.6557\alpha + 15.5612)/(0.61\alpha + 3.47)$.
$FWA(q_4) <=5.08 + 1.03\alpha$ for $78/105 <= \alpha <= 0.93436^*$
$FWA(q_4)$ is the minimum for $78/105 <= \alpha <= 0.93436^*$.

Subinterval [130/131, 163/164]: Ordering $[A_1,A_3,A_4,A_2,A_5,A_6,A_8,A_7]$.
$FWA(q_2) = (1.5259\alpha^2 + 9.7195\alpha + 13.493)/(1.12\alpha + 2.96)$.
$FWA(q_2) >= 4.56 + 1.29\alpha$ for $130/131 <= \alpha <= 163/164$.
$FWA(q_3) = (1.2937\alpha^2 + 9.1309\alpha + 14.3138)/(0.94\alpha + 3.14)$.
$FWA(q_3) <= 3.78+2.34\alpha$ for $130/131 <= \alpha <= 163/164$.
$FWA(q_3)$ is the minimum for $130/131 <= \alpha <= 163/164$.

Subinterval [163/164, 1]: Ordering $[A_1,A_3,A_4,A_2,A_6,A_5,A_8,A_7]$.
$FWA(q_2) = (1.5259\alpha^2 + 9.7195\alpha + 13.493)/(1.12\alpha + 2.96)$.
$FWA(q_2) >= 4.56 + 1.29\alpha$ for $163/164 <= \alpha <= 1$.
$FWA(q_3) = (1.2937\alpha^2 + 9.1309\alpha + 14.3138)/(0.94\alpha + 3.14)$.
$FWA(q_3) <= 3.78 + 2.34\alpha$ for $163/164<=\alpha<=1$.
$FWA(q_3)$ is the minimum for $163/164 <= \alpha <=1$.

Putting the pieces together, we have found that the minimum is

$(0.7537\alpha^2 + 9.2443\alpha + 14.7404)/(0.79\alpha + 3.29)$. if $0 <= \alpha <= 0.19463^*$.
$(0.5215\alpha^2 + 8.6557\alpha + 15.5612)/(0.61\alpha + 3.47)$. if $0.19463^* <= \alpha <= 0.93436^*$.
$(1.2937\alpha^2 + 9.1309\alpha + 14.3138)/(0.94\alpha + 3.14)$. if $0.93436^* <= \alpha <= 1$

Next we calculate the maximum of the α-cuts of the fuzzy weighted average. There are 2 intersections of right-hand sides: A_2 and A_6 intersect at 136/245 and A_4 and A_6 intersect at 61/171..

Subinterval [0,61/171]: Ordering $[A_1,A_3,A_6,A_4,A_2,A_5,A_8,A_7]$.
$FWA(q^7) = (-1.8193\alpha^2 + 4.4713\alpha + 22.0864)/(0.98\alpha + 3.1)$.
$FWA(q^7) <= 10 - 0.74\alpha$ for $0 <= \alpha<= 61/171$.
$FWA(q^6) = (-1.6269\alpha^2 + 1.6789\alpha + 24.6864)/(0.72\alpha + 3.36)$.
$FWA(q^6) <= 9.59 - 2.37\alpha$ for $0 <= \alpha <= 61/171$.
$FWA(q^5) = (-0.75\alpha^2 - 2.7463\alpha + 28.2347)/(0.35\alpha + 3.73)$.
$FWA(q^5) <= 8.63 - 2.51\alpha$ for $0 <= \alpha <= 61/171$.
$FWA(q^4) = (0.0783\alpha^2 - 6.4225\alpha + 31.0826)/(0.02\alpha + 4.06)$.
$FWA(q^4) <= 7.88-1.77\alpha$ for $0 <= \alpha <=61/171$.
$FWA(q^3) = (0.2553\alpha^2 - 7.3875\alpha + 31.8706)/(-0.08\alpha + 4.16)$.
$FWA(q^3) >= 7.27 - 0.06\alpha$ for $0 <= \alpha <= 0.25062^*$.
$FWA(q^3)$ is the maximum for $0 <= \alpha <= 0.25062^*$.
$FWA(q^3) <= 7.27 - 0.06\alpha$ for $0.25062^* <= \alpha <= 61/171$.
$FWA(q^2) = (0.2835\alpha^2 - 10.8326\alpha + 35.2875)/(-0.55\alpha + 4.63)$.
$FWA(q^2)>= 7.17-1.32\alpha$ for $0.25062^* <= \alpha <= 61/171$.
$FWA(q^2)$ is the maximum for $0.25062^* <= \alpha <= 61/171$.

Subinterval [61/171,136/245]: Ordering $[A_1,A_3,A_4,A_6,A_2,A_5,A_8,A_7]$.
$FWA(q^7) = (-1.8193\alpha^2 + 4.4713\alpha + 22.0864)/(0.98\alpha + 3.1)$.
$FWA(q^7) <= 10 - 0.74\alpha$ for $61/171 <= \alpha <= 136/245$.
$FWA(q^6) = (-1.6269\alpha^2 + 1.6789\alpha + 24.6864)/(0.72\alpha + 3.36)$.
$FWA(q^6) <= 9.59-2.37\alpha$ for $61/171 <= \alpha <= 136/245$.
$FWA(q^5) = (-0.75\alpha^2 - 2.7463\alpha + 28.2347)/(0.35\alpha + 3.73)$.
$FWA(q^5) <= 8.63 - 2.51\alpha$ for $61/171 <= \alpha <= 136/245$.
$FWA(q^4) = (0.0783\alpha^2 - 6.4225\alpha + 31.0826)/(0.02\alpha + 4.06)$.
$FWA(q^4) <= 7.27 - 0.06\alpha$ for $61/171 <= \alpha <= 136/245$.
$FWA(q^3) = (0.1065\alpha^2 - 9.8676\alpha + 34.4995)/(-0.45\alpha + 4.53)$.
$FWA(q^3) <= 7.88 - 1.77\alpha$ for $61/171 <= \alpha <= 136/245$.
$FWA(q^2) = (0.2835\alpha^2 - 10.8326\alpha + 35.2875)/(-0.55\alpha + 4.63)$.
$FWA(q^2) >= 7.17-1.32\alpha$ for $61/171 <= \alpha <= 136/245$.
$FWA(q^2)$ is the maximum for $61/171 <= \alpha <= 136/245$.

Subinterval [136/245,1]: Ordering $[A_1,A_3,A_4,A_2,A_6,A_5,A_8,A_7]$.
$FWA(q^7) = (-1.8193\alpha^2 + 4.4713\alpha + 22.0864)/(0.98\alpha + 3.1)$.
$FWA(q^7) <= 10 - 0.74\alpha$ for $136/245 <= \alpha <= 1$.
$FWA(q^6) = (-1.6269\alpha^2 + 1.6789\alpha + 24.6864)/(0.72\alpha + 3.36)$.
$FWA(q^6) <= 9.59 - 2.37\alpha$ for $136/245 <= \alpha <=1$.
$FWA(q^5) = (-0.75\alpha^2 - 2.7463\alpha + 28.2347)/(0.35\alpha + 3.73)$.
$FWA(q^5) <= 7.27 - 0.06\alpha$ for $136/245 <= \alpha <= 1$.

$FWA(q^4) = (-0.7218\alpha^2 - 6.1914\alpha + 31.6516)/(-0.12\alpha + 4.2)$.
$FWA(q^4) <= 8.63 - 2.51\alpha$ for $136/245 <= \alpha <= 1$.
$FWA(q^3) = (0.1065\alpha^2 - 9.8676\alpha + 34.4995)/(-0.45\alpha + 4.53)$.
$FWA(q^3) <= 7.88 - 1.77\alpha$ for $136/245 <= \alpha <= 1$.
$FWA(q^2) = (0.2835\alpha^2 - 10.8326\alpha + 35.2875)/(-0.55\alpha + 4.63)$.
$FWA(q^2) >=7.17 - 1.32\alpha$ for $136/245 <= \alpha <= 1$.
$FWA(q^2)$ is the maximum for $136/245 <= \alpha <= 1$.

Putting the pieces together, we have found that the maximum is

$(0.2553\alpha^2 - 7.3875\alpha + 31.8706)/(-0.08\alpha + 4.16)$ if $0<=\alpha<=0.25062^*$,
$(0.2835\alpha^2 - 10.8326\alpha + 35.2875)/(-0.55\alpha + 4.63)$ if $0.25062^*<=\alpha<=1$.

This leads to the following membership function for the fuzzy weighted average:

$\mu(x) = 0$ if $x <= 4.480^*$
$\mu(x) = 0.524^*x - 6.133^* + 0.663^*\sqrt{0.6241x^2 - 4.687^*x + 41.018^*}$ if $4.480^* <= x <= 4.811^*$
$\mu(x) = 0.585^*x - 8.299^* + 0.959^*\sqrt{0.3721x^2 - 3.322^*x + 42.460^*}$ if $4.811^* <= x <= 5.966^*$
$\mu(x) = 0.363^*x - 3.529^* + 0.386^*\sqrt{0.8836x^2 - 0.917^*x + 9.302^*}$ if $5.966^* <=x <= 6.063^*$
$\mu(x) = -0.970^*x + 19.105^* - 1.764^*\sqrt{0.3025x^2 - 6.665^*x + 77.329^*}$ if $6.063^* <= x <= 7.255^*$
$\mu(x) = -0.157^*x + 14.468^* - 1.958^*\sqrt{0.0064x^2 + 3.066^*x + 22.029^*}$ if $7.255^* <= x <= 7.661^*$
$\mu(x) = 0$ if $x >= 7.661^*$

5 Conclusions

A lot of research effort has been invested into the development of algorithms for the calculation of fuzzy weighted averages. Where the computational complexity of the algorithms improved in the course of the time, leading to the linear algorithm of Guu [6], the approach has always been to compute the α-cuts of the fuzzy weighted average for fixed value of α. As a consequence, one can only compute a finite number of values of the membership function of the fuzzy weighted average. In this paper we have presented an algorithm for the computation of the membership function of the fuzzy weighted average analytically, for triangular or trapeizoidal weights and attributes. Our approach has been to generalise a simple, but not optimally efficient, single-α algorithm. Using our algorithm, one no longer needs to approximate the membership function from a finite number of values. The feasibility of our algorithm has been demonstrated by the explicit calculation of the exact membership functions of the fuzzy weighted averages of examples from the literature.

References

1. Chang, P.-T., Hung, K.-C., Lin, K.-P., Chang, C.-H.: A Comparison of Discrete Algorithms for Fuzzy Weighted Average. IEEE Transactions on Fuzzy Systems 14, 663–675 (2006)
2. Chiao, K.–P.: Direct Fuzzy Weighted Average Algorithm for Fuzzy Multiple Attributes Decision Making. Tamsui Oxford Journal of Mathematical Sciences 16, 311–327 (2000)
3. Dong, W.M., Wong, F.S.: Fuzzy Weighted Averages and Implementation of the Extension Principle. Fuzzy Sets and Systems 21, 183–199 (1987)
4. Guh, Y.–Y., Hon, C.–C., Lee, E.S.: Fuzzy Weighted Average: The Linear Programming Approach via Charnes and Cooper's Rule. Fuzzy Sets and Systems 117, 15–160 (2001)
5. Guh, Y.–Y., Hon, C.–C., Wang, K.–M., Lee, E.S.: Fuzzy Weighted Average: A Max–Min Paired Elimination Method. Computers Math. Applic. 32, 115–123 (1996)
6. Guu, S.–M.: Fuzzy Weighted Average Revisited. Fuzzy Sets and Systems 126, 411–414 (2002)
7. Kao, C., Liu, S.–L.: Fractional Approach to Fuzzy Weighted Average. Fuzzy Sets and Systems 120, 435–444 (2001)
8. Lee, D.H., Park, D.: An Efficient Algorithm for Fuzzy Weighted Average. Fuzzy Sets and Systems 87, 39–45 (1997)
9. Liou, Y.–C., Guu, S.–M.: Linear–time Algorithm for the Fuzzy Weighted Average Method. Journal of the Chinese Institute of Industrial Engineers 19, 7–12 (2002)
10. Liou, T.–S., Wang, M.–J.: Fuzzy Weighted Average: An Improved Algorithm. Fuzzy Sets and Systems 49, 307–315 (1992)
11. Broek, P.M., van den Noppen, J.A.R.: Fuzzy Weighted Average: Alternative Approach. In: 25th International Conference of the North American Fuzzy Information Processing Society (NAFIPS 2006). IEEE Computer Society Press, Los Alamitos (2006)

Part II
Evolutionary Computation

Knowledge-Based Constrained Function Optimization Using Cultural Algorithms with an Enhanced Social Influence Metaphor

Mostafa Ali[1], Robert Reynolds[2], Rose Ali[3], and Ayad Salhieh[1]

[1] Computer Information Technology, Jordan University of Science & Technology 22110, Irbid, Jordan
[2] Computer Science Department, Wayne State University, Detroit, 48202, Michigan, U.S.A.
[3] Computer Design, Yarmouk University, 21163, Irbid, Jordan
{mzali,salhieh}@just.edu.jo, Reynolds@cs.wayne.edu
rosa@yu.edu.jo

Abstract. In this research we present a new framework based on Cultural Algorithms using an enhanced social fabric influence function to solve nonlinearly constrained global optimization problems. We identify how knowledge sources used by Cultural Algorithms are combined to direct the decisions of the individual agents during the problem solving process using an influence function family based upon a Social Fabric metaphor. Guided interactions between the population swarms and these knowledge sources produced emergent phases of problem solving. This implies that the social interaction of individuals coupled with their interaction with a culture within which they are embedded provides a powerful vehicle for the solution of these problems. Results demonstrate that this approach can successfully extract interesting emergent patterns in the Belief space and improve the search efficiency by avoiding local Optima, and converge to an approximate global minimizer asymptotically. Different parameter combinations can affect the rate of solution.

Keywords: Constrained global optimization, Cultural algorithms, Cultural swarms, Knowledge swarms, Social evolution.

1 Introduction

The Cultural Algorithm (CA) is a class of computational models derived from observing the cultural evolution process in nature. It is a dual inheritance system that characterizes evolution in human culture at both the macro-evolutionary level, which takes place within the Belief space, and at the micro-evolutionary level, which occurs at the population space. Knowledge produced in the population space at the micro-evolutionary level is selectively accepted or passed to the Belief space and used to adjust the symbolic structures there. This knowledge can then be used to influence the changes made by the population in the next generation. The basic Cultural Algorithms framework is illustrated in Figure 1.

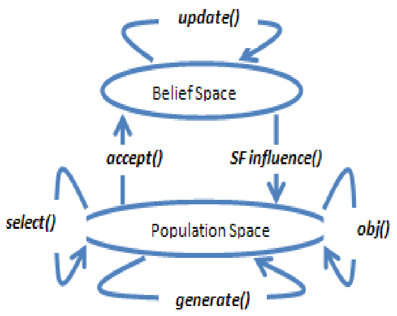


Fig. 1. The framework of cultural algorithm

Previous work by Reynolds [1] identified five basic categories of knowledge that were useful in decision making. They were normative knowledge (ranges of acceptable behaviors), situational knowledge (exemplars of successful and unsuccessful solutions), domain knowledge (knowledge of domain objects, their relationships, and interactions), history knowledge (temporal patterns of behavior), and topographical knowledge (spatial patterns of behavior). This set of categories is viewed as being complete for a given domain in the sense that all available knowledge can be expressed in terms of one of these classifications.

Reynolds [1] looked at the roles and contribution of these five generic knowledge classes (normative, topographical, domain, situational, and history knowledge) to the optimization problem-solving process using Evolutionary Programming (EP) as the population model. They observed the emergence of certain problem solving phases in terms of the relative performance of different knowledge sources over time. They labeled these phases as coarse grained, fine grained, and backtracking phases. Each phase is characterized by the dominance of a suite or subset of the knowledge sources that are most successful in generating new solutions in that phase. In fact, the dominant subset of knowledge sources is often applied in a specific sequence within each phase. It appears that one knowledge source produces new solutions that are consequently exploited in by another knowledge source. Transitions between phases occur when the solutions produced by one phase can be better exploited by knowledge sources associated with the next phase. These phases emerged in static, dynamic, and deceptive problem environments.

The coarse grained phase often dominates at the beginning of the search process or when the problem solving landscape changes dynamically, and a search for a new solution must begin anew. In the coarse grained phase topographical knowledge dominates, producing the best new solution over 50% of the time. Situational knowledge is the second most successful, producing the best new solution over 25% of the time. In the fine-grained phase situational knowledge is the most successful at generating the best new individual, while Normative and Domain knowledge are a distant second best. In the backtracking phase all of the knowledge sources are equally successful at generating new solution. Static problems are the exception, in which cases the history component has little effect. Likewise, in non-deceptive environments backtracking occurred less frequently than the other two phases.

Cultural Algorithms can provide a flexible framework in which to study the emergence of complexity in a multi-agent system (MAS) [2]. In this scenario the Cultural Algorithms framework has been embedded within the recursive porous agent simulation tool (Repast) [3], producing a toolkit that is called Cultural Algorithms Toolkit (CAT). This tool is used to view the power Cultural Algorithms in solving many Engineering problems and other type of problems [4].

While many successful real-world applications of Cultural Algorithms have been produced, we are interested in studying the fundamental computational processes involved the use of Cultural Systems as problem solvers. In previous work the influence of the knowledge sources have been on individuals in the population only. The goal of this paper is to examine how Cultural Algorithms solve nonlinearly constrained global optimization problems. In our investigation here, we employ a set of standard test problems with differentiable objective functions [reference]. These test problems are considered diverse enough to cover many kinds of difficulties that constrained global optimization faces. Agents then interact socially via the various knowledge sources to find the optimum after weaving the social fabric to motivate interaction. We then investigate the emergence of social patterns in both the population space and the Belief space when the problem is successfully solved.

In this new approach, the Social Fabric influence function is the key to finding the optimal for a certain minimization problem. The agents are connected through a topology that determines connectivity type between agents, through which the fabric is weaved after the initial signal is sent from the Knowledge Sources.

2 Cultural Algorithms

The basic pseudocode of Cultural Algorithm is given as follow:

```
Begin
    $t = 0;$
    initialize $B^t$, $P^t$
    repeat
        evaluate $P^t$
            update($B^t$, accept($P^t$))
            generate($P^t$, influence($B^t$))
            $t = t + 1;$
            select $P^t$ from $P^{t-1}$
    until (termination condition achieved)
End
```

Here P^t represents the Population component at time t, and B^t for the Belief Space at time t. The algorithm begins by initializing both the Population and the Belief Space. Then it enters the evolution loop until the termination condition is satisfied.

At the end of each generation, individuals in the Population Space are first evaluated with a performance function *obj()*. An acceptance function, *accept()* is then used to determine which individuals will be allowed to update the Belief Space. Experiences of those chosen elite individuals are then added to the contents of the Belief Space via function *update()*. Next, knowledge from the Belief space is allowed to influence the selection of individuals for the next generation of the population through the *influence()* function. The two feedback paths of information, one through the *accept()* and *influence()* functions, and the other through individual experience and the *obj()* function create a system of dual inheritance of both population and Belief.

The CA repeats this process for each generation until the pre-specified termination condition is met. In this way, the population component and the Belief space interact with and support each other, in a manner analogous to the evolution of human culture.

2.1 Knowledge Sources

Five basic categories of cultural knowledge have been identified: Normative Knowledge (NN), Situational Knowledge (SN), Domain Knowledge (DN), History Knowledge (HN), and Topographical Knowledge (TN).

Normative Knowledge is a set of promising variable ranges that provide standards for individual behaviors and guidelines within which individual adjustments can be made [5]. Normative Knowledge leads individuals to "jump into the good range" if they are not already there.

Situational Knowledge provides a set of exemplary cases that are useful for the interpretation of specific individual experience. Situational Knowledge leads individuals to "move toward the exemplars".

Topographical Knowledge was originally proposed to reason about region-based functional landscape patterns [6]. The whole landscape is divided into cells according to spatial characteristics and each cell keeps track of the best individual in its region. Topographical Knowledge guides individuals to emulate the cell-best (similar to local optima).

Domain Knowledge uses knowledge about the problem domain to guide search. For example, in a functional landscape composed of cones, knowledge about cone shape and the related parameters will be useful in reasoning about them during the search process.

Historical or Temporal Knowledge monitors the search process and records important events in the search. These important events may be either a significant move in the search space or a detection of landscape change. Individuals guided by the History Knowledge can consult those recorded events for guidance in predicting a move direction or shift of system resources.

The five categories of knowledge that have been identified in CA systems were added at different times to achieve different problem-solving capabilities [1] [5] [6]. Reynolds has suggested that this set of knowledge sources is complete in that any cultural knowledge can be expressed as some combination of the five. When woven together, they show interesting collective behaviors regarding different roles in the search process which we call Cultural Swarm.

These five knowledge sources are used with only minor modifications from the Cones-World problem. This rather seamless transition was another interesting and rather unexpected event. It suggests to us that these five knowledge sources provide sufficient coverage to deal with broad classes of problems.

2.2 Communication Protocols

The acceptance function selects the individual experiences that will be used to update the Belief space at each generation. Here, the acceptance function will take the experiences of the top individuals in the population and use that information to update all of the knowledge structures.

Individuals are updated using the same framework as in [7]. The key here is that the knowledge sources communicate with each other through the update process. That is, if one knowledge source produces a new high performing individual, that individual can be used to potentially adjust each of the other knowledge sources. This allows other sources to exploit this new knowledge.

The acceptance function determines which individuals and their behaviors can impact the Belief space knowledge. It is often determined as a percentage of the number of current individuals ranging between 1% and 100% of the population size, based upon selected parameters such as performance. For example, we can select the best performers (e.g. top 10%), worst performers (e.g. bottom 10%), or any combinations.

Also, a modified dynamic acceptance function can be used by changing the number of accepted individuals over time, using the following function

$$accept = top\left[p\% + \left\lfloor (\frac{p\%}{k}) \right\rfloor \right] \tag{1}$$

where p represents the percentage of the population space that will affect the Belief space; k is the number of time steps or generations in the current environment, and is reset to one with every environmental change.

The idea is illustrated in Figure 2, where the number of accepted individuals is doubled when k equals 1 in the above function. As the k increase the number of accepted individuals decreases. For example, if p in the acceptance function above is set to 20%, the number of the accepted individuals in the first generation (where k equals 1) will be 40% of the population space. In the second generation (when k equals 2), the number of accepted individuals is 30% of the population space.

The choice of influence function has a great impact on the problem solving process. Some influence functions are more successful than others, as measured by the success of the agents that each has influenced in the past. Early influence functions randomly applied the five knowledge sources to individuals in the population in order to guide their problem solving process. However, it was felt that some systematic application of the knowledge sources would be beneficial to the problem solving process.

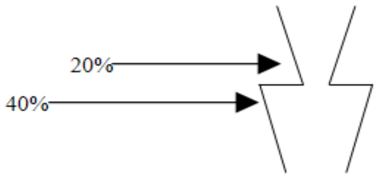


Fig. 2. Changing acceptance function

The influence function determines how the knowledge sources can influence the population. Here, when an individual is to be modified, the social fabric influence function interferes at the end of each generation and is weaved starting from the initial signal sent to the individuals, and ends up with the bidding procedure that happens between the individuals based on the constructed topology of interaction. Thus, as the population moves through the search landscape some knowledge sources may become more successful and others less successful.

2.3 The Cultural Algorithms Toolkit

Cultural Algorithms [8] can provide a flexible framework in which to study the emergence of complexity in a multi-agent system (MAS) [3]. In this scenario the Cultural Algorithms framework has been embedded within the recursive porous agent simulation tool (Repast) [3], producing a toolkit that is called Cultural Algorithms Toolkit (CAT) [4].

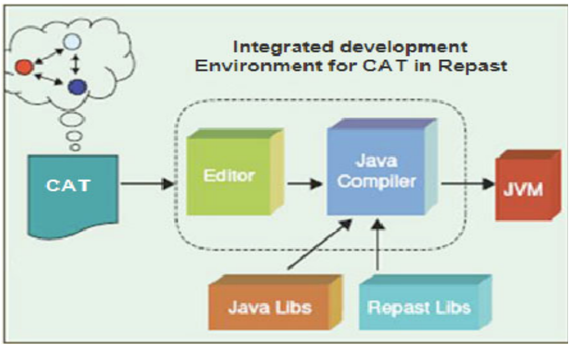


Fig. 3. The basic organization of the implementation of the CAT system in the Repast Integrated development Environment

This software tool is used to investigate the ability of Cultural Algorithms to solve many Engineering design problems among others [9]. Figure 3 gives the software architecture of how CAT is implemented in the basic Repast Integrated Development Environment.

3 Related Work

Several researchers have used different types of Algorithms for solving constrained optimization problems. A quick overview is as follows:

Coello and Mezura [10] implemented a version of the Niched-Pareto Genetic Algorithm (NPGA) [11] to handle constraints in single-objective optimization problems. The NPGA is a multiobjective optimization approach in which individuals are selected through a tournament based on Pareto dominance. However, unlike the first NPGA, Coello and Mezura's approach does not require niches (or fitness sharing [10]) to maintain diversity in the population. The NPGA is shown to be a more efficient technique than traditional multi-objective optimization algorithms, because it only uses a sample of the population to estimate Pareto dominance.

Deb [12] proposed a Genetic Adaptive Search (GeneAS) to solve engineering optimization problems. He uses both, binary and real encodings for each solution. This approach was tested on three engineering problems [12], emphasizing problems that have discrete and continuous variables. The obvious drawback of the approach is the need of implementing combined operators for the special encoding adopted which can be complex.

Mezura-Montes [10] presented an enhanced Evolutionary Algorithm that doesn't require the definition of extra parameters other than those used by the Evolutionary strategy. The implemented mechanism allows the algorithm to maintain diversity during the process. Reynolds [14] implemented an algorithm that uses the Marginal Value Theorem (MVT) to influence the individuals in the population and drive the process of obtaining better solutions. The algorithm was a more efficient one than the one presented in [1] [2].

4 The Social Fabric Influence Function

In this section we present the concept and mechanism through which the social fabric influence function can be used to direct the decisions of the individual agents in problem solving.

4.1 Concept

Knowledge sources are allowed to influence individuals through a network. From a theoretical perspective we view individuals in the real world as participating in a variety of different networks. Several layers of such networks can be supported within a population. The interplay of these various network computations is designated as the "social fabric". This notion of social fabric has appeared metaphorically in various ways within Computer Science. For example, IBM among others developed tools to reinforce the "social fabric" whereby designers and programmers interact to solve complex problems [17].

We adapt the Brock-Durlauf model of interactive discrete choice [18] to arbitrary interaction topologies represented by an arbitrary adjacency matrix $\mathbf{\Gamma}$*:* All individuals face the binary choice set $S = \{-1, 1\}$*:* Let agent i choose ω_i, $\omega_i \in S$, so as to maximize her utility, which depends on the actions of her neighbors: $Ui = U(\omega_i, \tilde{\omega}_{v(i)})$, where $\tilde{\omega}_{v(i)}$ denotes the vector of dimension d_i containing as elements the decisions made by each of agent i's neighbors, $j \in v(i)$. The I-vector of all agents' decisions, $\tilde{\omega} = (\omega_1, \dots, \omega_i)$*;* is also known as a *configuration,* and $\tilde{\omega}_{v(i)}$ is known as agent i's *environment.* We assume that an agent's utility function Ui is additively separable in a private utility component, which without loss of generality (due to the binary nature of the decision) may be written as $h\omega_i$, $h > 0$*;* in a social interactions component, which is written in terms of quadratic interactions between her own decision and of the expectation of the decisions of her neighbors, $\tilde{\omega}_{v(i)}$, $A\omega_i\varepsilon_i\{\frac{1}{|v(i)|}\sum_{j\in v(i)} J_{ij}\omega_j\}$*;* and a random utility component, $\epsilon(\omega_i)$*;* which is observable only by the individual i.

The social fabric is viewed as a computational tool that influences the action and interaction of the various knowledge sources. Informally, we have N networks and M individuals. An individual can be associated with one or more networks. For a given network only certain information is allowed to flow along that network between nodes. Each network can be viewed as being produced by a single thread that links up the participating nodes.

4.2 Weaving the Social Fabric into the CAT System

The networks that comprise the social fabric can emanate from either the Belief Space or from individuals in a network within the Population Space. In terms of the population, the network could reflect a kinship network or an economic network for example. In terms of the Belief Space, the network could be the Internet, or a local area network, or some other network directly accessible to the knowledge sources. It may be that the Knowledge Sources know something about the networks that they can access but are not sure how those networks are linked up to the low level social networks of the population. In other words, they may be aware of the outer layer of the social fabric, but can only infer about what is in the interior lining.

From the standpoint of the Knowledge Sources they can seed or influence a subset of individuals in the population, and that subset may have population level effects but the knowledge sources can only guess what they might be. The key is to “seed” a subset of the population so that they can affect other networks within a population.

As a simple example configuration in CAT we can simply specify just one network, one that is accessible to the Knowledge Sources in the Belief Space. What

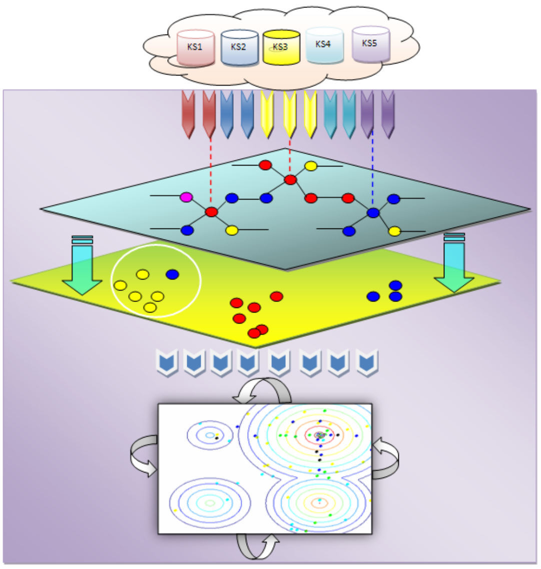


Fig. 4. Embedded the Social Fabric component in the Cultural Algorithms Toolkit with activated dynamics in the environment

we wish to investigate is whether just having access to the Social Fabric is sufficient for the Knowledge Sources to improve the performance of the influence function as opposed to not having a network to distribute their influence at all.

The process is illustrated in figure 4 where each individual first will be influenced by one knowledge source (as a special case) that will represent the initial signal to be passed to other individuals. The signal is passed to adjacent individuals in the topology based on the network connectivity. The individual is represented as a node in the landscape, where the number of connections or hops over which it can transmit this information to its neighbors corresponds to its influence. The maximum number of hops can be either 0 or d meaning either no connections or d connections at a time. The simplest case is configured by assuming that each individual is connected to a fixed number of other individuals using a constant topology. The topologies that we used here were taken from work in Particle Swarm Optimization where the impact of various topologies on the communication of local information among particles has been studied.

Several frequently used topologies taken from the Particle Swarm Optimization literature are supported in CAT. For example, the *lBest* model is the simplest form of a local topology is known as the ring model. The *lBest* ring model connects each individual to only two other individuals in the landscape and is shown in figure 5(a). Another frequently used topology is the *gBest* topology. In this topology each individual in the network is connected to all the individuals in the network as shown in figure 5(b). The advantage of the *lBest* model may lie in its lower convergence rate relative to the *gBest* model which may reduce the change of premature convergence to a false peak.

Another topology supported in CAT is the square topology in which each individual has four connections in addition to other individuals in the population.

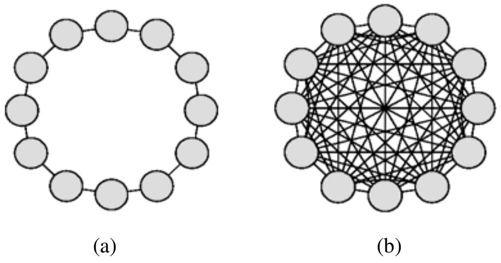


Fig. 5. Topologies used in the Social Fabric model for connection between individuals. (a) *lBest* ring topology. (b) *gBest* topology.

At each time step, every individual is influenced by one of the knowledge sources. In this simplest version, Knowledge Sources (KS) do not know anything about the network and the selected individuals' position in it. The individual then transmits the name of the influencing Knowledge Source to its neighbors through as many hops as specified. Next, each node counts up the number of Knowledge Source bids that it collects. The Knowledge Source that has the most votes is the winner and will direct the individual for that time step. This can be viewed as a weighted majority vote situation. This approach was used in Reynolds early Cultural Algorithms [19].

In case of a tie in the voting process, there are several tie breaking rules implemented in CAT. They include, select the "most frequently used KS, "the least frequently used Knowledge Source", and "the Knowledge Source that selected the individual this time", among others.

5 Experimental Framework and Results Analysis

In this section the results of application of the new social influence function, the Social Fabric, to different problems with varying complexity in terms of dimensions, and nonlinear constraints. Results are compared with other well-known algorithms from literature.

5.1 Experimental Framework

The number of individuals is fixed to 100. If a tie is found the resolution approach used is to use the Knowledge Source that directly affected the individual at that step.

The algorithm will be tested on a set of standard test problems G1-G13 [18] [20] [21] [22] except G2, since the objective function of problem G2 is not differentiable. These test problems are considered diverse enough to cover many kinds of difficulties that constrained global optimization faces [18] [23] [24], and have been used to test performances of algorithms for constrained global optimization.

The algorithm was used to solve each problem 30 times with 100 individual agents in the population space. The number of generations is set to a maximum of 15000 generations which is for problems G1 and G13. We experimented with different kinds of topologies through which we found that the best was the *lBest* topology.

Throughout the next subsection, we will use problem G4 to illustrate in detail how Cultural Algorithms solve such constrained optimization problems efficiently.

5.2 Analysis of Results

The approach used by Reynolds in [14] did not assume that there is any kind of connection between the individuals in the population space. Knowledge sources will pass their signals to the individuals at each time step. Our approach uses different topologies to pass abstract information obtained from the Knowledge Sources and then weave the social fabric to allow the individuals to pass the received info through the assumed used topology. The amount of interaction appears to affect the way the system solves the presented problem of a certain complexity. Not only the individual follows the successful Knowledge Source but also tries to adapt through neighbors in the built network to find a better value in the landscape.

To understand how the technique works, we will discuss the details of a drill-down through problem G4, which has 5 dimensions and nonlinear complex constraints, and is stated as follows:

$$\min_{x} f(x) = 5.3578547x_3^2 + 0.8356891x_3x_5 + 37.293239x_1 - 40792.141$$

s.t.

$$\begin{aligned}
g_1(x) &= u(x) - 92 \le 0,\\
g_2(x) &= -u(x) \le 0,\\
g_3(x) &= v(x) - 110 \le 0\\
g_4(x) &= -v(x) + 90 \le 0,\\
g_5(x) &= w(x) - 25 \le 0,\\
g_6(x) &= -w(x) + 20 \le 0,
\end{aligned}$$

Where

$$\begin{aligned}
u(x) &= 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5,\\
v(x) &= 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2,\\
w(x) &= 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4\ .
\end{aligned}$$

The bounds:

U = (102, 45, 45, 45, 45) and L = (78, 33, 27, 27, 27).

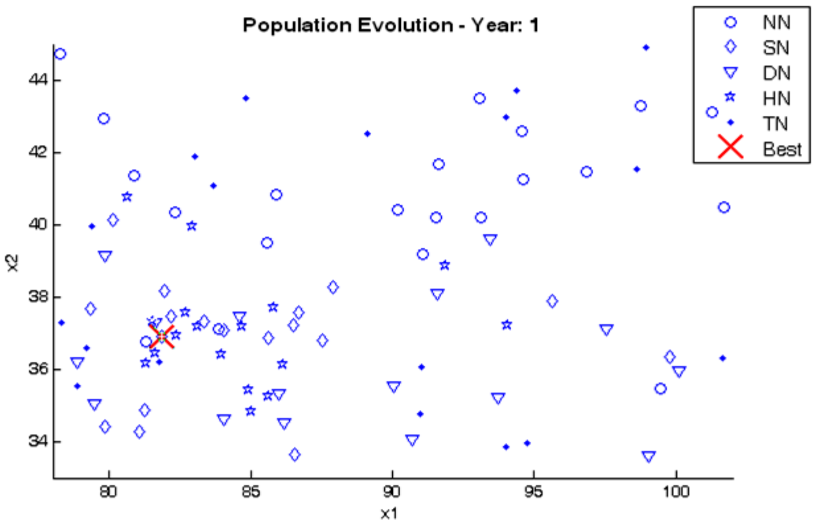
Global minimum:

x^* = (78, 33, 29.995256025682, 45, 36.775812905788), $f(x^*)$ = –30665.539.

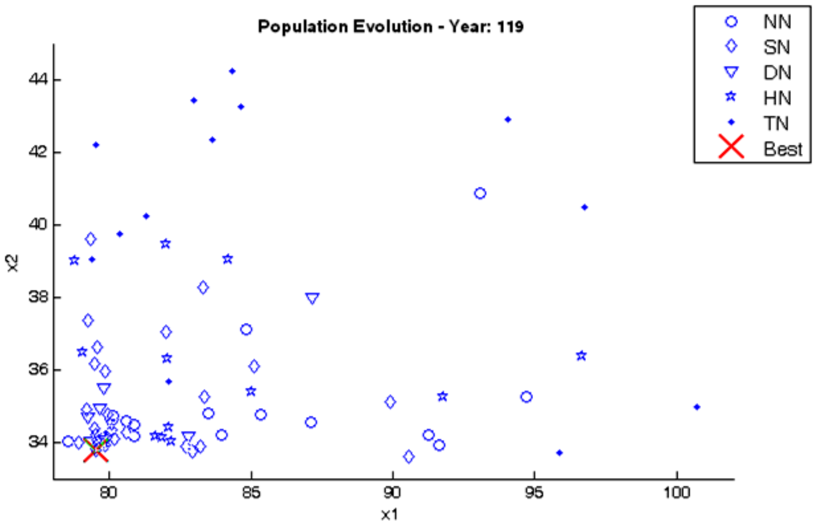
The population swarm plots in Figures 6-a and 6-b show the population (individuals) moving within the problem's constructed landscape using the *lBest* topology used by our Social Fabric (SF) approach. Each individual is shape coded to reflect the knowledge source that has influences it in that generation. The best individual of a generation is stressed using an 'X'. Since the results of the dimensions of problems can be explained similarly we discuss only dimensions x1 and x2. Figure 7 shows a sample of the constructed Social Fabric-*lBest* topology for problem G4.

Figures 6-a and 6-b show the initial generation and generation 119 when running the system using the Social Fabric-*lBes*t to illustrate how the different knowledge sources work under the influence of the social fabric technique to control individuals. The Topographic Knowledge followers draw the fine-tuning knowledge followers: Situational, Normative, and most of the Domain Knowledge followers. By generation 119 most of the individuals are swarming around the best. Topographic knowledge individuals are still exploring the space hoping to find a better solution to report it later to the fine-tuning knowledge followers.

The power behind the algorithm lies in using the bounding boxes that the system calculates at each time step for each of the Knowledge Sources as illustrated in figure 8. A bounding box represents the standard deviation of each "dot" produced during that generation for the mutation process. It is considered to be the focus of the generation process by each knowledge source. The main idea is how these bounding boxes of the Knowledge Sources interact (overlap area), and how focused these bounding boxes are at each time step. The branching phase of the algorithmic process is shown in Figures 8-a and 8-b, where initially the bounding boxes associated with the Topographic and



(a)



(b)

Fig. 6. Population swarm of dimension x1+x2 using the *lBest* topology. (a) Plotted at generation 1. (b) Plotted at generation 119.

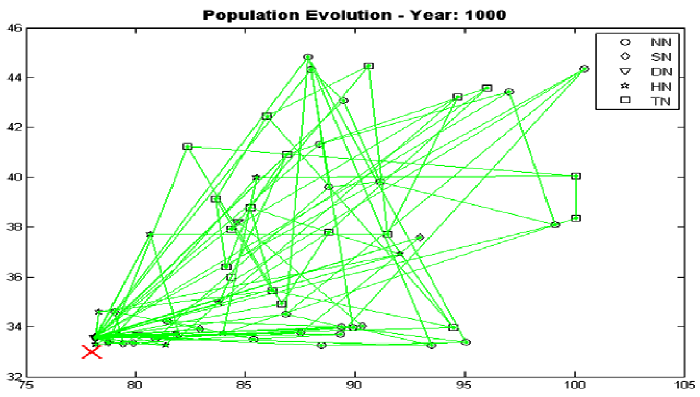
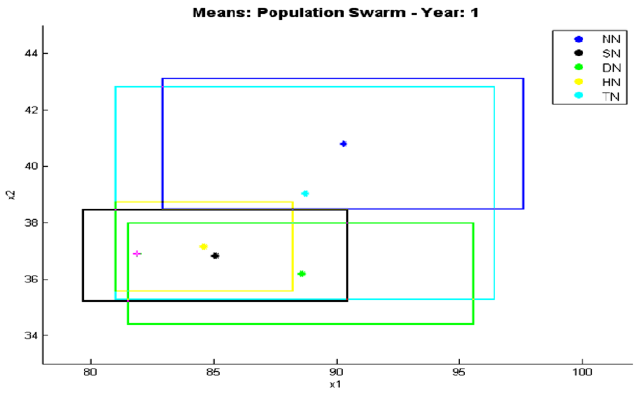
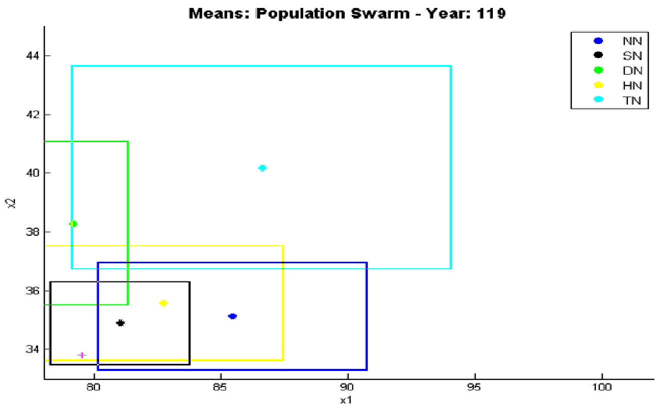


Fig. 7. A sample Social Fabric swarm plot for problem G4 using *lBest* topology



(a)

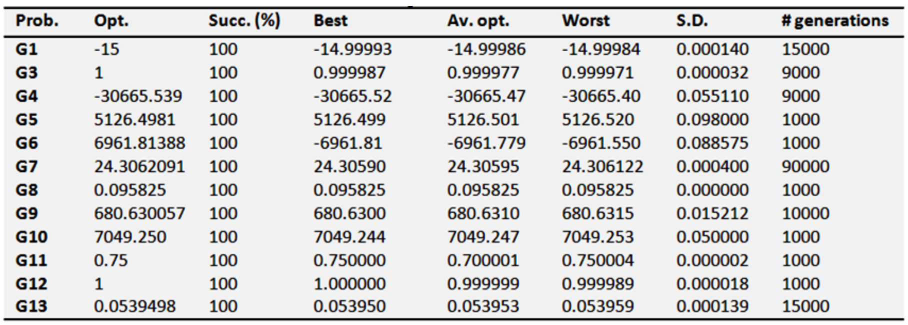


(b)

Fig. 8. Knowledge Swarm Plot of dimension x1+x2. (a) Plotted at generation 1. (b) Plotted at generation 119.

Normative Knowledge Sources cover most of the space. The exploitation process takes place with time and the bounding boxes for the fine-grained search process have separated from those for the coarse-grained phase (focused search vs. wider search) and have surrounded the optimal value for this pair of dimensions. These bounding boxes are effectively channeling new individuals into this area as can be seen in figure 8-b.

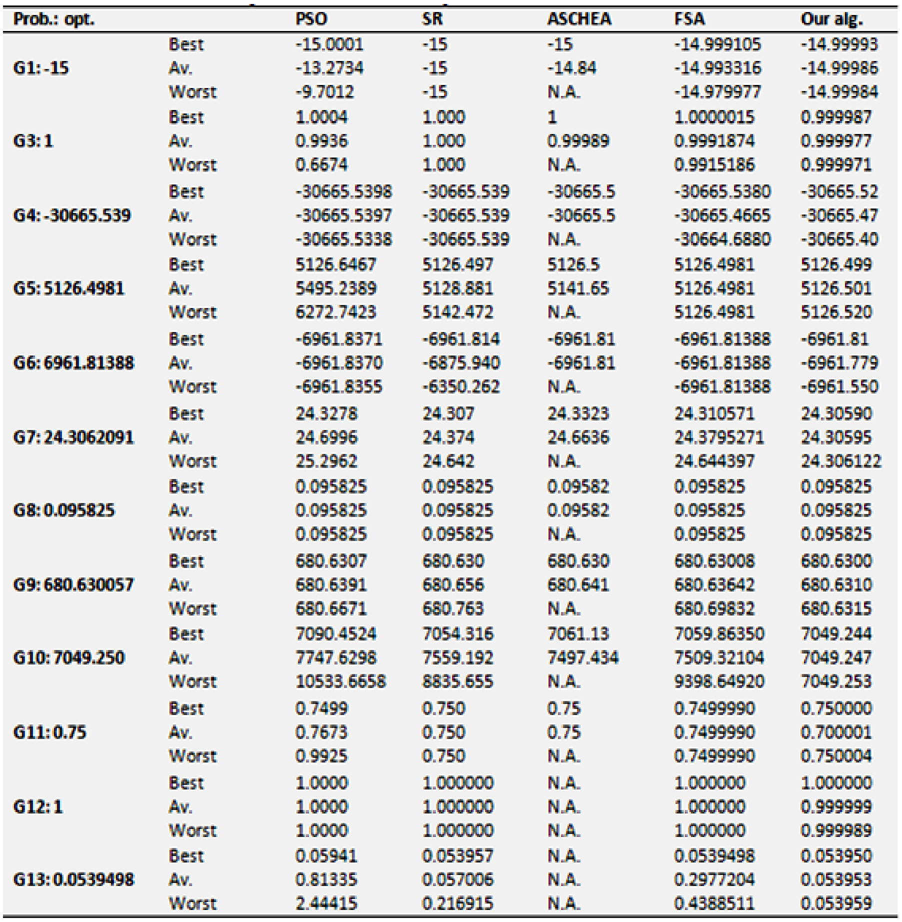
Table 1. Tests results for problems G1-G13



Prob.	Opt.	Succ. (%)	Best	Av. opt.	Worst	S.D.	# generations
G1	-15	100	-14.99993	-14.99986	-14.99984	0.000140	15000
G3	1	100	0.999987	0.999977	0.999971	0.000032	9000
G4	-30665.539	100	-30665.52	-30665.47	-30665.40	0.055110	9000
G5	5126.4981	100	5126.499	5126.501	5126.520	0.098000	1000
G6	6961.81388	100	-6961.81	-6961.779	-6961.550	0.088575	1000
G7	24.3062091	100	24.30590	24.30595	24.306122	0.000400	90000
G8	0.095825	100	0.095825	0.095825	0.095825	0.000000	1000
G9	680.630057	100	680.6300	680.6310	680.6315	0.015212	10000
G10	7049.250	100	7049.244	7049.247	7049.253	0.050000	1000
G11	0.75	100	0.750000	0.700001	0.750004	0.000002	1000
G12	1	100	1.000000	0.999999	0.999989	0.000018	1000
G13	0.0539498	100	0.053950	0.053953	0.053959	0.000139	15000

Table 1 shows the best known objective function value in the second column. We report in Table 1 the best and the worst optimal values obtained from 30 runs for each test problem. To understand quality of the obtained solutions, the average optimal value, and the standard deviation of the obtained objective function values for all 30 runs are given. Moreover, the success rate, the maximum number of generations before we stop each run, used to obtain these results in 30 runs, are reported in the last columns two of Table 1 for each problem respectively. The results in table 2 show a statistical comparison between our new approach and some other known approaches from literature. When plotting the population swarms, individuals are plotted in different shapes to indicate which knowledge source is in control of that individual at that time step.

Table 2. Comparison of test results for problems G1-G13



Prob.: opt.		PSO	SR	ASCHEA	FSA	Our alg.
	Best	-15.0001	-15	-15	-14.999105	-14.99993
G1: -15	Av.	-13.2734	-15	-14.84	-14.993316	-14.99986
	Worst	-9.7012	-15	N.A.	-14.979977	-14.99984
	Best	1.0004	1.000	1	1.0000015	0.999987
G3: 1	Av.	0.9936	1.000	0.99989	0.9991874	0.999977
	Worst	0.6674	1.000	N.A.	0.9915186	0.999971
	Best	-30665.5398	-30665.539	-30665.5	-30665.5380	-30665.52
G4: -30665.539	Av.	-30665.5397	-30665.539	-30665.5	-30665.4665	-30665.47
	Worst	-30665.5338	-30665.539	N.A.	-30664.6880	-30665.40
	Best	5126.6467	5126.497	5126.5	5126.4981	5126.499
G5: 5126.4981	Av.	5495.2389	5128.881	5141.65	5126.4981	5126.501
	Worst	6272.7423	5142.472	N.A.	5126.4981	5126.520
	Best	-6961.8371	-6961.814	-6961.81	-6961.81388	-6961.81
G6: 6961.81388	Av.	-6961.8370	-6875.940	-6961.81	-6961.81388	-6961.779
	Worst	-6961.8355	-6350.262	N.A.	-6961.81388	-6961.550
	Best	24.3278	24.307	24.3323	24.310571	24.30590
G7: 24.3062091	Av.	24.6996	24.374	24.6636	24.3795271	24.30595
	Worst	25.2962	24.642	N.A.	24.644397	24.306122
	Best	0.095825	0.095825	0.09582	0.095825	0.095825
G8: 0.095825	Av.	0.095825	0.095825	0.09582	0.095825	0.095825
	Worst	0.095825	0.095825	N.A.	0.095825	0.095825
	Best	680.6307	680.630	680.630	680.63008	680.6300
G9: 680.630057	Av.	680.6391	680.656	680.641	680.63642	680.6310
	Worst	680.6671	680.763	N.A.	680.69832	680.6315
	Best	7090.4524	7054.316	7061.13	7059.86350	7049.244
G10: 7049.250	Av.	7747.6298	7559.192	7497.434	7509.32104	7049.247
	Worst	10533.6658	8835.655	N.A.	9398.64920	7049.253
	Best	0.7499	0.750	0.75	0.7499990	0.750000
G11: 0.75	Av.	0.7673	0.750	0.75	0.7499990	0.700001
	Worst	0.9925	0.750	N.A.	0.7499990	0.750004
	Best	1.0000	1.000000	N.A.	1.000000	1.000000
G12: 1	Av.	1.0000	1.000000	N.A.	1.000000	0.999999
	Worst	1.0000	1.000000	N.A.	1.000000	0.999989
	Best	0.05941	0.053957	N.A.	0.0539498	0.053950
G13: 0.0539498	Av.	0.81335	0.057006	N.A.	0.2977204	0.053953
	Worst	2.44415	0.216915	N.A.	0.4388511	0.053959

6 Conclusions

The Cultural Algorithm (CA) is a stochastic optimization method that uses evolutionary algorithmic mechanisms to model cultural evolution and social behaviors. Just as cultural evolution contributes to the adaptability of human society, CA provides an additional degree of adaptability to evolutionary computation. In this paper we have introduced the social fabric influence (SFI) function in the Cultural Algorithms framework. This influence function produces population and knowledge swarms that are used to optimally solve nonlinearly constrained optimization problems. The Social Fabric metaphor allows the knowledge sources to distribute their influence through a social network.

We applied this approach to a set of well-known nonlinearly constrained optimization problems. It turns out that the topology employed, frequency of distribution of influence, and the conflict resolution mechanisms selected play an important role in how efficiently the system produces knowledge and population swarms for a given problem.

In future work we will investigate the construction of hierarchically structured social fabrics for use within the solution of a suite of problem presented to the system in a simultaneous fashion.

References

1. Reynolds, R., Saleem, S.: The Impact of Environmental Dynamics on Cultural Emergence. In: Festschrift, in Honor of John Holland, pp. 1–10. Oxford University Press, Oxford (2003)
2. Reynolds, R.: A Metrics-Based System to Monitor the Stepwise Refinement of Program Modules. In: Fourth Conference on Intelligent Systems and Machines, April 29-30, Oakland University (1986)
3. North, M., Collier, N., Vos, J.: Experiences Creating Three Implementations of the Repast Agent Modelling Toolkit. ACM Transactions on Modelling and Computer Simulation 16(1), 1–25 (2006)
4. Reynolds, R., Ali, M.: Exploring knowledge and population swarms via an agent-based Cultural Algorithms Simulation Toolkit (CAT). In: IEEE Congress on Evolutionary Computation CEC 2007, September 25-28, pp. 2711–2718 (2007)
5. Chung, C., Reynolds, G.R.: CAEP: An Evolution-based Tool for Real-Valued Function Optimization using Cultural Algorithms. International Journal on Artificial Intelligence Tools 7(3), 239–291 (1998)
6. Jin, X., Reynolds, G.R.: Using Knowledge-Based Evolutionary Computation to Solve Nonlinear Constraint Optimization Problems: a Cultural Algorithm Approach. In: Proceeding of the 1999 Congress on Evolutionary Computation, pp. 1672–1678. IEEE Press, Washington DC (1999)
7. Rychtyckyj, N., Ostrowski, D., Schleis, G., Reynolds, G.R.: Using Cultural Algorithms in Industry. In: Proceedings of IEEE Swarm Intelligence Symposium. IEEE Press, Indianapolis (2003)
8. Reynolds, R.G.: An Introduction to Cultural Algorithms. In: Proceedings of the 3rd Annual Conference on Evolutionary Programming, pp. 131–139. World Scientific Publishing, Singapore (1994)
9. Reynolds, R.G., Ali, M.Z.: Embedding a Social Fabric Component into Cultural Algorithms Toolkit for an Enhanced Knowledge-Driven Engineering Optimization. International Journal of Intelligent Computing and Cybernetics 1(4), 563–597 (2008)
10. Coello, C., Mezura-Montes, E.: Handling Constraints in Genetic Algorithms Using Dominance-Based Tournaments. In: Parmee, I. (ed.) Proceedings of the Fifth International Conference on Adaptive Computing Design and Manufacture (ACDM 2002), University of Exeter, Devon, UK, vol. 5, pp. 273–284. Springer, Heidelberg (2002)
11. Horn, J., Nafpliotis, N., Goldberg, D.: A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In: Proceedings of the First IEEE Conference on Evolutionary Computation, WCCI, vol. 1, pp. 82–87. IEEE Service Center, Piscataway (1994)
12. Deb, K., Goldberg, D.: An Investigation of Niche and Species Formation in Genetic Function Optimization. In: Schaffer, J.D. (ed.) Proceedings of the Third International Conference on Genetic Algorithms, pp. 42–50. Morgan Kaufmann Publishers, San Mateo (1989)

13. Deb, K., Goyal, M.: A Combined Genetic Adaptive Search GeneAS for Engineering Design. Computer Science and Informatics 26(4), 30–45 (1996)
14. Reynolds, R., Peng, B.: Cultural algorithms: computational modeling of how cultures learn to solve problems: an engineering example. Cybernetics and Systems 36(8), 753–771 (2005)
15. Coello, C.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. Computer Methods in Applied Mechanics and Engineering 191, 1245–1287 (2002)
16. Coelho, L., Souza, R., Mariani, V.: Improved differential evolution approach based on cultural algorithm and diversity measure applied to solve economic load dispatch problems. Mathematics and Computers in Simulation 79(10) (2009)
17. Cheng, L., Patterson, J., Rohall, S., Hupfer, S., Ross, S.: Weaving a Social Fabric into Existing Software. In: AOSD 2005: Fourth International Conference on Aspect-Oriented Software Development, Chicago, IL, RC23485 (2005)
18. Brock, W.A., Durlauf, S.N.: Discrete Choice with Social Interactions. Review of Economic Studies 68(2), 235–260 (2001)
19. Reynolds, R.G.: On Modeling the Evolution of Hunter-Gatherer Decision-Making Systems. Geographical Analysis 10(1), 31–46 (1978)
20. Hedar, A., Fukushima, M.: Derivative-free filter simulated annealing method for constrained continuous global optimization. Journal of Global Optimization 35, 521–549 (2006)
21. Hock, W., Schittkowski, K.: Test Examples for Nonlinear Programming Codes. Springer, Heidelberg (1981)
22. Koziel, S., Michalewicz, Z.: Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. Evolutionary Computation 7(1), 19–44 (1999)
23. Michalewicz, Z., Schoenauer, M.: Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation 4(1), 1–32 (1996)
24. Wenxing, Z., Ali, M.: Solving nonlinearly constrained global optimization problem via an auxiliary function method. Journal of Computational and Applied Mathematics (2009)

Reconstructing Dynamic Target Functions by Means of Genetic Programming Using Variable Population Size

Leonardo Vanneschi[1] and Giuseppe Cuccu[2]

[1] Dipartimento di Informatica, Sistemistica e Comunicazione (D.I.S.Co.)
University of Milano-Bicocca, Milan, Italy
[2] Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Lugano, Switzerland

Abstract. Dynamic environments are becoming more and more popular in many applicative domains. A large amount of literature has appeared to date dealing with the problem of tracking the extrema of dynamically changing target functions, but relatively few material has been produced on the problem of reconstructing the shape, or more generally finding the equation, of dynamically changing target functions. Nevertheless, in many applicative domains, reaching this goal would have an extremely important impact. It is the case, for instance, of complex systems modelling, like for instance biological systems or systems of biochemical reactions, where one is generally interested in understanding what's going on in the system over time, rather than following the extrema of some target functions. Last but not least, we also believe that being able to reach this goal would help researchers to have a useful insight on the reasons that cause the change in the system over time, or at least the pattern of this modification. This paper is intended as a first preliminary step in the attempt to fill this gap. We show that genetic programming with variable population size is able to adapt to the environment modifications much faster (i.e. using a noteworthy smaller amount of computational effort) than standard genetic programming using fixed population size. The suitability of this model is tested on a set of benchmarks based on some well known symbolic regression problems.

1 Introduction

Many real-world problems are anchored in dynamic environments, where some element of the problem domain, typically the target, changes with time. For this reason, developing solid evolutionary algorithms (EAs) to reliably solve these problems is an important task. The application of evolutionary computation (EC) to dynamic environments creates challenges different to those encountered in static environments. Foremost among these, are issues of premature convergence, and the evolution of overfit solutions. In the last few years, many contributions have appeared which studied dynamic optimization environments and developped new evolutionary frameworks for solving them. Some of those contributions are discussed in Section 2. Nonetheless, the majority of those approaches are based on Genetic Algorithms (GAs) [17] or Particle Swarm Optimization (PSO) [8] and the problem objective is to find the extrema (maxima or minima) of a target function that changes with time. On the other hand, very few contributions have appeared to date that study the ability of Genetic Programming (GP) [22] to reconstruct target functions on dynamic optimization environments.

In this paper we hypothesize that variable size population GP is a promising method for dynamic optimization problems. This idea is not new in evolutionary computation; for instance, it has been applied to PSO in [13]. However, it has never been applied to GP before. We propose a variable size population GP model called *DynPopGP*. It is inspired by the one presented in [28]. Simply speaking, it works by shrinking the population when fitness is improving and increasing its size, by adding new genetic material, when the evolution stagnates. Our hypothesis is that when the target function changes, evolution of the current population should stagnate. (because it has been evolved to approximate the previous target function). Thus the evolution should benefit from the creation of new genetic material, that should give the necessary amount of diversity to start the optimization of a new target function.

Simply speaking, the motivation for this hypothesis is the following: assume the optimization environment we want to study is characterized by a dynamic target function that, at different time intervals can assume the value of one of the functions belonging to a given set $\{f_1, f_2, ..., f_n\}$. Let f_1 be the target function at time interval 1. Suppose GP is executed for some generations using f_1 as target function. It is likely that the GP popultion will loose diversity (see for instance [7]) and individuals will tend to specialize for approximating f_1. Assume now that the target function changes and becomes f_2. According on *how different* f_2 is from f_1, the GP population may have problems in re-adapting for the optimization of f_2. We hypothesize that in many cases re-initializing the population with random individuals would be a better strategy than trying to optimize f_2 with a population specialized for f_1. But of course the population can be re-initialized only if we assume that we know the instant when the target function changes from f_1 to f_2, which is clearly not acceptable in practice. For this reason, we develop a variable size GP framework that should adapt the population size with the following principle: we decrease the population size when the algorithm is progressing (i.e. fitness is improving) and we increase it, adding new individuals, when fitness does not improve anymore. In this way, when the target function changes from f_1 to f_2, we count on having a small population composed by individuals specialized for f_1. This population will be then filled with random individuals, thus tending towards a situation that approximates a re-initialization of the population. This re-initialization would be automatic and the proposed framework does not assume any knowledge of the instants when the target function changes.

We test our hypothesis on a set of symbolic regression dynamic test functions, discussed in Section 4.

This paper is structured as follows: in Section 2 we discuss previous contributions in dynamic optimization. In Section 3 we discuss the reasons why it is not suitable to directly apply the GP model presented in [28] to dynamic optimization and we present *DynPopGP* that extends it. Section 4 contains a description of the test problems and presents the experimental setting used in this paper. In Section 5 we show the obtained experimental results. Finally, Section 6 concludes the paper and suggests directions for future research.

2 Dynamic Optimization, Previous and Related Work

Most research on EC focuses on optimization of static, non-changing, problems. Many real-world optimization problems, however, are actually dynamic, and optimization methods capable of continuously adapting the solution to a changing environment are needed. The main problem with standard EAs used for dynamic optimization problems appears to be that EAs eventually converge to an optimum and thereby loose their diversity necessary for efficiently exploring the search space and consequently also their ability to adapt to a change in the environment when such a change occurs.

Over the past few years, a number of authors have addressed the problem of EAs premature loss of diversity in dynamic environments in many different ways. Surveys of these studies can be found for instance in [2,3,4,5]. Interestingly, in [3] Branke proposes a classification of these approaches into four broad categories: (1) Approaches that run EAs in a standard fashion, but, as a change in the environment is detected, take actions to increase diversity and thus facilitate the shift to the new optimum [9,30]. (2) Approcahes that reinject diversity in the population at all the time, hoping that this can allow the EA to adapt to changes more easily [19,23]. (3) Approaches that supply EA with memory, able to recall useful information from past generations, which seems especially useful when the target repeatedly returns to previous locations [18,24,10]. (4) Approaches that split the EA population in multiple subpopulations, some to track known local optima, some to search for new optima [6,29].

Contributions that belong to category (1) are for instance [9], where an hypermutation operator is defined to drastically increase diversity for some number of generations and a variant thereof, called *variable local search* presented in [30]. Contributions that belong to category (2) include [19], where the concept of *random immigrants* (randomly generated individuals that enter the population at each generation) is introduced and [23], where a new approach called *termodynamical GA* is presented. Contributions that use memory-based approaches (category (3)) include, for instance, studies on multipolidy, like, among many others [18,24], other redundant representation schemes like the one presented in [10], or the work of Branke [2]. Contributions that belong to category (4) are for instance [6], where so called *self-organizing-scouts* are introduced, or [29], where a new model of multi-population GA, called *forking GA*, is presented. More recent contributions include [31] where, based on the concept of problem difficulty, a new dynamic environment generator using a decomposable trap function is proposed; [20], where a coevolutionary agent-based model is used; [26] where the use of mutation for diversity maintenance is investigated and [11], where the use of artificial immune networks for multimodal function optimization on dynamic environments is studied.

All the above quoted contributions treat the problem of tracking the extrema in a dynamic environment, where the target function changes with time and concern GAs, PSO or other EAs variants. Very few contributions have appeared to date dealing with the (more complex) problem of approximating/reconstructing target functions that change with time by means of GP. Noteworthy recent exceptions are: [12], where financial time series (index closing price data) are reconstructed by means of Grammatical Evolution, and [27] where an approach for incorporating learning probabilistic context-sensitive grammar in GP is employed for the evolution and adaptation of locomotion gaits of a simulated snake-like robot. Nevertheless, both these approaches use Grammar-Based GP and employ it for very particular and complex applications.

The goal of this paper is different: first of all, we want to study standard tree-based GP [22], and one variant thereof using variable size populations; secondly, we want to present and employ (here for the first time) more simple, and thus easier to study, test problems. The proposed GP framework is presented in Section 3 and the used test functions in Section 4.

3 Variable Size Population GP

In 2003, an idea for counteracting the negative effects of *bloat* [1,25] and of premature convergence [7] on GP was presented. It consisted in reducing the size of populations at a linear rate [16,15]. This was achieved by removing a fixed number of individuals at each generation. This technique was called *plague* and it has been shown to have some positive effects on GP systems. That idea started from the observation of a general behavior of GP over a wide set of problems: normally fitness improves quickly at the beginning of GP runs and, after a number of generations, improvements are more difficult to obtain. In this second phase, plagues allow to save computational effort, that would be wasted otherwise, since it does not bring appreciable advantages.

On the other hand, it is clear that even if a considerable amount of computational effort is saved, the blind deletion of individuals at each generation probably cannot lead to the discovery of better individuals than the ones found by the standard GP process. Furthermore, steadily decreasing populations produce a progressive loss of diversity, especially at the genotypic level. For this reason, in [28], an extension of the plague technique aimed at varying the population size in an intelligent way during the execution of each GP run, was presented. In that model, adds and suppressions of individuals are operated dynamically on the basis of the behavior of the GP system: population size is decreased while the algorithm is progressing (i.e. fitness is improving) and it is increased when the algorithm reaches the stagnation phase. In this way, when the algorithm is progressing, as much computational effort as possible is saved and this previously saved effort is spent only when it is really useful, i.e. when the algorithm is stagnating and new genetic material is needed. In [28] the decision whether to shrink or inflate the population was taken on the basis of the relationship between the best fitness value in the population at the current generation g (bf_g) and the one at the previous generation (bf_{g-1}). This value was stored in a variable the authors called *pivot*. Two versions of *pivot* are presented in [28]: in the first one *pivot*$= \Delta_{g-1}/\Delta_g$ and in the second one *pivot*$= \Delta_{g-1} - \Delta_g$, where $\Delta_g = bf_{g-1} - bf_g$. The GP model using the first version of pivot was called DIV, while the one using the second version was called SUP in [28].

In Section 3.1 we discuss the reasons why DIV and SUP are not suitable to solve dynamic optimization problems and in Section 3.2 we present our new variable size population GP model, called *DynPopGP*, that extends DIV and SUP.

3.1 DIV and SUP in Dynamic Environments

Both the DIV and SUP methods introduced in [28] have the following characteristics:

(i) The decision on whether to shrink or inflate the population is taken only on the basis of the relationship between the best fitness values at the current generation and at the previous one. This decision does *not* depend on how good those fitness values are. In other words, this decision is the same independently from the fact that GP has found good solutions or bad ones.

(ii) The quantity of individuals that have to be added to or suppressed from the population depends on the current population size (in both DIV and SUP when individuals have to be suppressed, 1% of the population is suppressed and when they need to be added, a number of individuals equal to the 0.2% of the population is added). Thus, additions and suppressions are more violent when the population is large.

A consequence of point (i) is that, when applied to dynamic optimization, DIV and SUP behave exactly the same in case the algorithm stagnates on a particular target function (but the target function remains the same) and in case the target function changes. However, when the target function changes, and in particular if the new target function is "different enough" from the old one, we expect a more violent worsening in fitness than when the algorithm stagnates on a fixed target function. In particular, if we use an elitist algorithm (i.e. we copy the best, or a pool of good individuals, unchanged in the next population at each generation), the best fitness in the population cannot worsen if the target function stays the same. It can only worsen if the target function has changed (because the best individual at the previous generation may change its fitness). The algorithm we propose (*DynPopGP*) behaves in two different ways in these two different situations.

A consequence of point (ii) is that, when the target function changes, the population size may grow indefinitely. In fact, suppose DIV or SUP are optimizing a given target function and are in a stagnation phase. Then, the population keeps growing. Now, assume the target function changes. The population will continue growing up until the new genetic material necessary to optimize the new target function has been created. *DynPopGP* solves this problem by defining a new function to quantify the amount of individuals that have to be added or deleted from a population.

For these reasons, we do not consider the DIV and SUP models any longer in this paper. A detailed experimentation showing the practical advantages of *DynPopGP* compared to DIV and SUP is definitely needed in the future.

3.2 New Variable Size Population Model

The *DynPopGP* algorithm we propose can be summarized by the pseudo-code in Figure 1, where we consider minimization problems (i.e. small fitness values are better than large ones)[1]. This algorithm uses a number of parameters (*trg_fit*, *old_got_trg*, *new_got_trg*, *stand_by_size*) and functions (*update_pop_size*, Δ_{pop}) that we describe below. Empirical values for these parameters, coming from a set of preliminary experiments, have been used in this work. More detailed sensitivity analysis on these parameters definitely deserves to be conducted in the future.

[1] The authors are aware that in case of minimization problems the term *fitness* is rather incorrect. Nevertheless, they keep using it for simplicity.

```
begin
   Generate a population of $N$ random individuals;
   $best$ = best individual in the population;
   $old\_got\_trg$ = false;
   for $g := 1$ to $maxgen$ do
       $new\_got\_trg$ = (fitness($best$) $\leq$ $trg\_fit$);
       if (not $new\_got\_trg$)
         then
           elitism (i.e. copy of the best);
           selection;
           reproduction / crossover;
           mutation;
           $best$ = best individual in the new population;
           $new\_got\_trg$ = (fitness($best$) $\leq$ $trg\_fit$);
           if ($old\_got\_trg$)
               then
                 // The old best had reached the target, while the new best has not reached it:
                 // the target function has surely changed. Set the population size to the initial size
                 $update\_pop\_size(N - current\_pop\_size)$;
               else
                 // Neither the new best, nor the old best have reached the target: update the
                 // population size using the $\Delta_{pop}$ function
                 $update\_pop\_size(\Delta_{pop}())$;
           endif
         else
             if (not $old\_got\_trg$)
             then
               // The new best has reached the target, while the old best had not reached it.
               // This means that the target has been found now. I have to spend as few
               // computational effort as possible until the target function changes (or the process.
               // terminates). I set the population size to a prefixed "stand-by" value
         $update\_pop\_size(stand\_by\_size - current\_pop\_size)$;
             endif
         endif
         $old\_got\_trg = new\_got\_trg$;
   endfor
end
```

Fig. 1. Pseudo-code for the *DynPopGP* algorithm

trg_fit represents a fitness value (target) that approximates the optimum. In this work, we have used a value equal to 0.01.

old_got_trg (respectively *new_got_trg*) is a boolean variable whose value is *true* if the best fitness in the population at the previous (respectively current) generation is better than or equal to the target and *false* otherwise.

stand_by_size is a small value of the population size that is used when the optimum of a target function has been approximated in a satisfaisable way, and we have to wait for the target function to change. In this case, the population has to be as small as possible, so that we can save computational effort. In our work, we wanted to set this value as a function of the initial population size. We have chosen a value of *stand_by_size* equal to the initial population size divided by 4 because, by means of a set of experiments, we have seen that this value represents a good compromise between saving computational effort (population shrinking) and keeping some good genetic material in the population.

update_pop_size(x) is a function that adds $|x|$ individuals to the population if x is positive and suppresses $|x|$ individuals from the population if x is negative. When $|x|$ individuals have to be suppressed, the population is sorted. The $2 \cdot x$ worst (in terms of fitness) individuals in the population are considered and, among these individuals, the x largest ones (in terms of number of tree nodes) are suppressed (as it happened for the DIV and SUP algorithms). When $|x|$ individuals have to be added, they are randomly generated with the same initialization method that is used at the beginning of the GP run (ramped half-and-half in this work).

$\Delta_{pop}()$ is a function that returns the number of individuals that have to be to be added or suppressed from the population when neither the old best fitness value nor the new one approximate the optimal fitness value in a satisfaisable way. We want $\Delta_{pop}()$ to be a function of the current best fitness in the population and the current population size, which are the two basic principles that were not taken into account by the DIV and SUP algorithms. For doing this, we define two new functions: *best_fit_contribution* and *pop_size_contribution* and we multiply their returned values. We want the result of this multiplication to be immediately interpretable by a human, so we impose that the results of *best_fit_contribution* and *pop_size_contribution* belong to the range $[1, 10]$. In this way, their product belongs to the range $[1, 100]$ and it can be interpreted, for instance, as a percentage (which represents the respective contributions given by the best fitness value and the population size). The $\Delta_{pop}()$ function performs the following calculation:

$$\begin{aligned}\Delta_{pop}() = &\, pivot \cdot strength \cdot best\_fit\_contribution() \cdot \\ &\, pop\_size\_contribution()\end{aligned}$$

where:

pivot is a variable whose value is -1 if the best fitness in the population at the current generation is better then the one at the previous generation and $+1$ otherwise (in practice, the value of *pivot* determines if individuals have to be added or suppressed).

strength is a variable that determines how strong populations inflate and deflate have to be at each step, and it is used to rescale the value of *best_fit_contribution*$() \cdot$ *pop_size_contribution*$()$. In this work, we use a value equal to 0.3. In this way, the maximum number of individuals that can be added to or suppressed from the population is 30 (given that the maximum possible value of *best_fit_contribution*$() \cdot$ *pop_size_contribution*$()$ is 100). In fact, experimental evidence confirms that adding more than 30 individuals at a time to the population eccessively increments the computational effort without a corresponding gain in the quality of the generated solutions.

The *best_fit_contribution*$()$ function determines the contribution given to the $\Delta_{pop}()$ by the best fitness value reached. As we have said above, we want this function to return a value in the range $[1, 10]$. Furthermore, we want it to return 10 (maximum contribution to the $\Delta_{pop}()$) when the best fitness in the population is bad (fitness above a certain threshold, 60 in this work) and to return the minimum value when the best fitness in the population approximates the optimum in a satisfaisable way. The easier way

```
$best\_fit\_contribution()$ ::
    if ( fitness($best$) $\leq trg\_fit$) then return $min\_coeff$;
    elsif (fitness($best$) $\geq max\_fit$) then return $max\_coeff$;
    else return $max\_coeff - min\_coeff \cdot \frac{fitness(best) - trg\_fit}{max\_fit - trg\_fit} + min\_coeff$
    endif
```

Fig. 2. Pseudo-code for the *best_fit_contribution* function

to obtain this, is to define the $best\_fit\_contribution()$ as a linear function, for instance a straight line, that intersects the points (trg_fit, min_coeff) and (max_fit, max_coeff) where *min_coeff* is equal to 1, *max_coeff* is equal to 10 and *max_fit* is equal to 60. So, the $best\_fit\_contribution()$ function is defined by the pseudo-code in Figure 2.

Finally, the $pop\_size\_contribution()$ function determines the contribution to the $\Delta_{pop}()$ given by the current population size. Analogously to the *best_fit_contribution*, we have used a linear function that returns the maximum possible value (10 in this work) when the current population size is minimal (i.e. it is equal to *stand_by_size*, that has been set to the initial population size divided by 4 in this work) and the minimum possible value (0 in this work) when the current population size is maximal (i.e. smaller or equal to *stand_by_size*).

4 Test Problems and Experimental Setting

Some benchmark problems have been defined for testing the performances of optimization methods in dynamic environments. In particular, Branke [4,5] defines and uses *moving peaks* types of functions. In these benchmarks, hand-tailored fitness landscapes are defined and the positions of the extrema and their basins of attraction are modified with time. Similar problems are also used, for instance, in [20,26,11].

However, this type of benchmark is not suitable for the present study. In fact, in this work, we want to study the ability of GP to reconstruct dynamic target functions and not follow moving extrema. With this goal in mind, it would make no sense to use moving peaks benchmarks as the ones presented in [4,5], given that, in those kinds of benchmark, extrema are moved by changing some additive or multiplicative constants to a (otherwise not changing) target function. If one uses GP with linear scaling (introduced in [21]), the moving peaks problem reduces to a static GP problem, given that linear scaling allows to reconstruct the shape of the target functions, offering a method to automatically determine additive and multiplicative constants.

For this reason, in this paper we define a new set of benchmark problems that can be used to test GP ability to reconstruct target functions in dynamic environments. These benchmarks are symbolic regression problems inspired by [21]. In particular, maintaining the same terminology as in [21], we have considered test functions F_{12}, F_{13}, F_{14}, F_{15} and F_{16} (presented in [21] at page 9) and we have used them to build dynamic test problems in which the importance of the modification of the target function can be tuned. Even though presented in [21], we also report here the equations for these functions: $F_{12}(x,y) = xy + sin((x-1)(y-1))$, $F_{13}(x,y) = x^4 - x^3 + y^2/2 - y$, $F_{14}(x,y) = 6\,sin(x)\,cos(y)$, $F_{15}(x,y) = 8/(2+x^2+y^2)$ and $F_{16}(x,y) = x^3/5 - y^3/2 - y - x$.

```
begin
  Define a set of test functions $F = \{f_1, f_2, ..., f_n\}$
  for $g := 1$ to $maxgen$ do
    For each fitness case $(x,y)$, the target value is: $\sum_{i=1}^{n} f_i(x,y)$;
    if ($g$ mod $period$ = 0) then $\forall 1 \le i \le n : f_i := succ(f_i)$ endif
  endfor
end
```

Fig. 3. Pseudo-code for target calculation in benchmark problems BENCH1, BENCH2 and BENCH3 The difference between these benchmark is in the size of set F: $n = 2$ for BENCH1; $n = 3$ for BENCH2 and $n = 4$ for BENCH3

As in [21], for all these functions the fitness cases are created by generating 20 random values (with uniform distribution) for x and y in the range $[-3,3]$.

We are aware that these test functions are bi-dimensional and thus do not represent real-life applications (typically characterized by many features and thus multi-dimensional), nevertheless, as reported in [21] at page 8: "The aim of this set of experiments is to demonstrate the practical implications of the use of the [method] studied here. Being of low dimensionality does not make the problems easy however. Many of the problems above mix trigonometry with polynomials, or make the problems in other ways highly non-linear".

Using these test functions, we have built three benchmarks for dynamic optimization that we have called BENCH1, BENCH2 and BENCH3. The target function at each generation is calculated by the algorithm in Figure 3, where given a test function F_i, with $12 \le i \le 15$ $succ(F_i) = F_{i+1}$ and $succ(F_{16}) = F_{12}$. The difference between these benchmaks is in the cardinality of the set of functions F used for calculating the target: for BENCH1, F contains two functions. These functions are F_{12} and F_{13} at generation 1. The target value is calculated performing the sum of these two functions for each couple of points (x,y). At each *period* generations, one of the two functions changes (i.e. it is deleted from set F and replaced by another function), while the other stays the same, in a cyclic way so that all the test functions are used.

BENCH2 is like BENCH1, except that F contains 3 functions, that are F_{12}, F_{13} and F_{14} at generation 1 and at each *period* generations, one of them changes, while the other two stay the same.

BENCH3 is similar, except that F contains 4 functions, that are F_{12}, F_{13}, F_{14} and F_{15} at generation 1 and at each *period* generations, one of them changes, while the other three stay the same.

In this way, BENCH1 has the more violent target modifications at each *period* generations, BENCH3 has the less violent modifications, while BENCH2 is in an intermediary situation.

In this work, we have used a value of *period* equal to 20. The other parameters used are as follows: population size of 200 individuals; function set equal to $\{+,-,*,/\}$ (exactly the same method as in [21] has been used to avoid divisions with denominator equal to zero and thus to ensure operators closure); terminal set composed by two floating point variables and four ephemeal random constants; maximum tree depth for

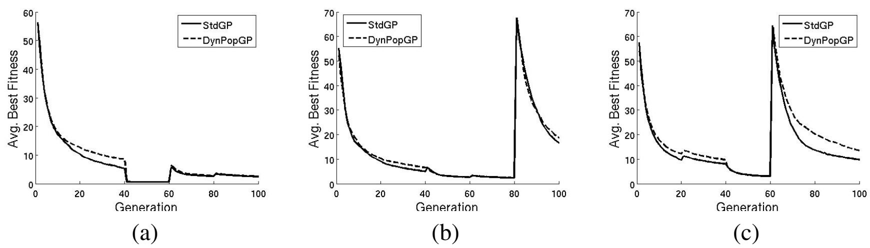


Fig. 4. Average best fitness against generations for *stdGP* and *DynPopGP*. (a): BENCH1; (b): BENCH2; (c): BENCH3.

initialization equal to 6; maximum tree depth for crossover and mutation equal to 17; tournament size equal to 10; standard subtree crossover [22] applied with probability 0.9; standard subtree mutation [22] applied with probability 0.1; maximum number of generations equal to 100 (in this way, given that *period*= 20, the process stops when the target function returns the same as at generation 1); generational GP with elitism (i.e. copy of the best individual unchanged in the next population at each generation). Fitness is the root mean squared error (RMSE) between outputs and targets. All the results reported in the next section have been obtained by performing 100 independent runs of each GP model (standard GP and *DynPopGP*) for each banchmark. With standard GP we indicate the canonic (fixed size population) GP process [22].

5 Experimental Results

In Figure 4 we report average best fitness values over 100 independent runs against generations for standard GP (*stdGP*) and *DynPopGP* for BENCH1 (Figure 4(a)), BENCH2 (Figure 4(b)) and BENCH3 (Figure 4(c)). This figure clearly shows that the two GP models find solutions of similar qualities at corresponding generations for all the three studied benchmarks (standard deviation error bars, not shown here for simplicity, confirm that the differences between the curves in Figure 4 are not statistically relevant). Seen from this perspective, the two GP models might seem equivalent. However, as reported for instance in [14], comparing the performances of two GP models against *generations* may lead to wrong conclusions, given that GP individuals have a variable size and thus evaluating a generation for the two models may request a very different amount of computational resources.

For this reason, in Figure 5 we report the values of the computational effort against generations (values averaged over the same 100 runs as in Figure 4). We have considered exactly the same definition of computational effort as in [14], i.e. the computational effort at a given generation g (E_g) is given by: $E_g = PE_g + PE_{g-1} + ... + PE_1$, where the partial effort at generation g (PE_g) is defined as the sum of the numbers of nodes of all the individuals in the population at generation g. Given that fitness calculation is often the most computationally expensive part of an EA and that in GP this calculation largely depends on the size of the individuals in the population, this measure clearly gives an idea of the computational complexity of executing a GP model (as claimed

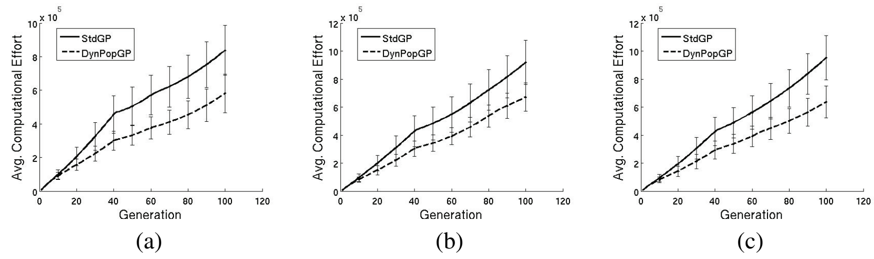


Fig. 5. Computational effort against generations for *stdGP* and *DynPopGP*. (a): BENCH1; (b): BENCH2; (c): BENCH3.

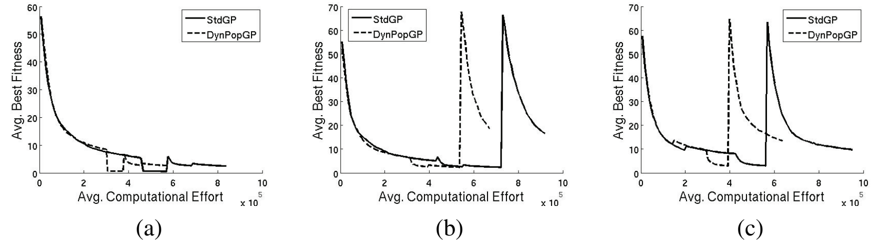


Fig. 6. Average best fitness against computational effort for *stdGP* and *DynPopGP*. (a): BENCH1; (b): BENCH2; (c): BENCH3.

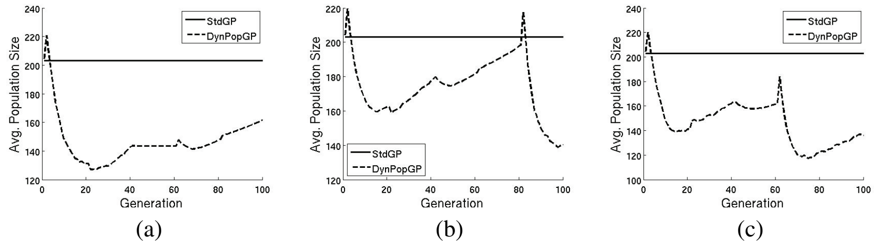


Fig. 7. Population size against generations for *stdGP* and *DynPopGP*. (a): BENCH1; (b): BENCH2; (c): BENCH3.

in [14]). Figure 5 shows that the effort spent by *DynPopGP* is smaller than the one spent by *stdGP* for all the three studied benchmarks. Standard deviations reported in figure as error bars seem to hint that these results are statistically significant.

Authors of [14] report results of the average best fitness against computational effort. We do the same in Figure 6, where it is clear that *DynPopGP* finds solutions of similar quality with a smaller computational effort than *stdGP*.

In Figure 7 we report the average population size at each generation (calculated using the same 100 runs as in the previous figures). We can see that the population size of *DynPopGP* is always smaller than the one of *stdGP* for all the three studied benchmarks. Nonetheless, we can notice that the population size of *DynPopGP*, tends to grow at each *period* generations (generation number multiple of 20), because of the modification in the target function. In some cases this growth begins slightly before the end of a period, probably because the particular target function had already been optimized and the stagnation phase was beginning. Another interesting thing to remark is that, after a first phase of population shrinking, which is common in the three benchmarks, the population growth is stronger for BENCH1 (which has the more violent target modifications) than for BENCH3 (which has the less violent target modifications), while the behavior of BENCH2 is intermediary. Furthermore, it is possible to see that for BENCH1 the population size continues to grow until the end of the run, while for BENCH2 and BENCH3 there is a new phase in which the population starts shrinking once again (at about generation 80 for BENCH2 and generation 60 for BENCH3).

6 Conclusions

This paper investigates the usefulness of variable size population Genetic Programming (GP) on dynamic problems. The idea is not new in Evolutionary Computation [13], but, to the best of our knowledge, it had never been extended to GP before. In particular, we believe that a model inspired by the one presented in [28] should be suitable for this kind of problems.

Contributions of this paper are: first of all, we have motivated the fact that the GP model presented in [28], taken as it is, is not suitable for dynamic optimization problems. Successively, we have presented a GP model that extends the one introduced in [28] and that we candidate for suitably solving dynamic optimizaton problems. We have called that model *DynPopGP*. We have also defined a new set of benchmarks to test GP models for dynamic optimization, based on some symbolic regression problems used in [21]. Finally, we have experimentally shown that *DynPopGP* allows GP to save computational effort compared to standard GP, while finding solutions of the same accuracy, at least for the studied benchmarks.

This work is clearly a first and preliminary step in this research track. The usefulness of GP (and in particular GP with variable size population) for dynamic optimization deserves further investigation. In particular, GP models have to be tested on hard real-life applications, typically characterized by a large number features and few samples and the issue of generalization to out-of-sample data deserves to be investigated.

References

1. Banzhaf, W., Langdon, W.B.: Some considerations on the reason of bloat. Genetic Programming and Evolvable Machines 3, 81–91 (2002)
2. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Congress on Evolutionary Computation CEC 1999, vol. 3, pp. 1875–1882. IEEE, Los Alamitos (1999)
3. Branke, J.: Evolutionary approaches to dynamic environments - updated survey. In: GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, pp. 27–30 (2001)

4. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer, Dordrecht (2001)
5. Branke, J.: Evolutionary approaches to dynamic optimization problems – introduction and recent trends. In: Branke, J. (ed.) GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, pp. 2–4 (2003)
6. Branke, J., Kauler, T., Schmidt, C., Schmeck, H.: A multi-population approach to dynamic optimization problems. In: Adaptive Computing in Design and Manufacturing, pp. 299–308. Springer, Heidelberg (2000)
7. Burke, E., Gustafson, S., Kendall, G., Krasnogor, N.: Advanced population diversity measures in genetic programming. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 341–350. Springer, Heidelberg (2002)
8. Clerc, M.: Particle Swarm Optimization. ISTE (2006)
9. Cobb, H.G.: An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical Report ADA229159, Naval Research Lab, Washington DC (1990)
10. Dasgupta, D., Mcgregor, D.R.: Nonstationary function optimization using the structured genetic algorithm. In: Parallel Problem Solving From Nature, pp. 145–154. Elsevier, Amsterdam (1992)
11. de França, F.O., Von Zuben, F.J., de Castro, L.N.: An artificial immune network for multimodal function optimization on dynamic environments. In: GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 289–296. ACM, New York (2005)
12. Dempsey, I.: Grammatical Evolution in Dynamic Environments. PhD thesis, University College Dublin, Ireland (2007)
13. Fernandes, C., Ramos, V., Rosa, A.C.: Varying the population size of artificial foraging swarms on time varying landscapes. In: Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S. (eds.) ICANN 2005. LNCS, vol. 3696, pp. 311–316. Springer, Heidelberg (2005)
14. Fernández, F., Tomassini, M., Vanneschi, L.: An empirical study of multipopulation genetic programming. Genetic Programming and Evolvable Machines 4(1), 21–52 (2003)
15. Fernández, F., Tomassini, M., Vanneschi, L.: Saving computational effort in genetic programming by means of plagues. In: Congress on Evolutionary Computation (CEC 2003), Canberra, Australia, pp. 2042–2049. IEEE Press, Piscataway (2003)
16. Fernández, F., Vanneschi, L., Tomassini, M.: The effect of plagues in genetic programming: A study of variable size populations. In: Ryan, C., et al. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 317–326. Springer, Heidelberg (2003)
17. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
18. Goldberg, D.E., Smith, R.E.: Nonstationary function optimization using genetic algorithms with dominance and diploidy. In: ICGA, pp. 59–68 (1987)
19. Grefenstette, J.J.: Genetic algorithms for changing environments. In: Parallel Problem Solving from Nature, vol. 2, pp. 137–144 (1992)
20. Huang, C.-F., Rocha, L.M.: Tracking extrema in dynamic environments using a coevolutionary agent-based model of genotype edition. In: GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 545–552. ACM, New York (2005)
21. Keijzer, M.: Improving symbolic regression with interval arithmetic and linear scaling. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 71–83. Springer, Heidelberg (2003)
22. Koza, J.R.: Genetic Programming. The MIT Press, Cambridge (1992)

23. Mori, N., Kita, H., Nishikawa, Y.: Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 513–522. Springer, Heidelberg (1998)
24. Ng, K.P., Wong, K.C.: A new diploid scheme and dominance change mechanism for non-stationary function optimization. In: Proceedings of the 6th International Conference on Genetic Algorithms, pp. 159–166. Morgan Kaufmann Publishers Inc., San Francisco (1995)
25. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming (2008), Published via, http://lulu.com, http://www.gp-field-guide.org.uk (With contributions by J. R. Koza)
26. Rand, W., Riolo, R.: The problem with a self-adaptative mutation rate in some environments: a case study using the shaky ladder hyperplane-defined functions. In: GECCO 2005: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 1493–1500. ACM, New York (2005)
27. Tanev, I.: Genetic programming incorporating biased mutation for evolution and adaptation of snakebot. Genetic Programming and Evolvable Machines 8(1), 39–59 (2007)
28. Tomassini, M., Vanneschi, L., Cuendet, J., Fernández, F.: A new technique for dynamic size populations in genetic programming. In: Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC 2004), Portland, Oregon, USA, pp. 486–493. IEEE Press, Piscataway (2004)
29. Tsutsui, S., Fujimoto, Y., Ghosh, A.: Forking genetic algorithms: Gas with search space division schemes. Evol. Comput. 5(1), 61–80 (1997)
30. Vavak, F., Jukes, K., Fogarty, T.C.: Learning the local search range for genetic optimisation in nonstationary environments. In: IEEE Intl. Conf. on Evolutionary Computation ICEC 1997, pp. 355–360. IEEE Publishing, Los Alamitos (1997)
31. Yang, S.: Constructing dynamic test environments for genetic algorithms based on problem difficulty. In: Congress on Evolutionary Computation, CEC 2004, vol. 2, pp. 1262–1269. IEEE, Piscataway (2004)

Interactive Evolution for Designing Motion Variants

Jonathan Eisenmann[1], Matthew Lewis[2], and Bryan Cline[1]

[1] Computer Science & Engineering, The Ohio State University
2015 Neil Ave, Columbus, OH, U.S.A.
[2] Advanced Computing Center for the Arts & Design, The Ohio State University
1224 Kinnear Rd, Columbus, OH, U.S.A.

Abstract. We present an intuitive method for novice users to interactively design custom populations of stylized, heterogeneous motion, from one input motion. The user sets up lattice deformers which are used by a genetic algorithm to manipulate the animation channels of the input motion and create new motion variants. Our interactive evolutionary design environment allows the user to traverse the available space of possibilities, presents the user with populations of motion, and gradually converges to a satisfactory set of solutions. Each generated motion can undergo a filtering process subject to user-specified, high-level metrics to produce a result crafted to fit the designer's interest. We demonstrate application to both character animation and particle systems.

Keywords: Evolutionary design, Animation, Interaction techniques, Crowds, Particle systems.

1 Introduction

The human visual system possesses acute pattern recognition abilities which can expose unnatural qualities in synthetic crowd animation if insufficient motion variation is present. Often times, crowd animations will also need to display particular meaningful expressive qualities in order to set the mood for a scene. The motions of crowd agents must not only be diverse and expressive, but they also often need to appear visually coherent. Therefore a certain amount of similarity between some of the individuals in a crowd is desirable.

A common method for achieving variation within a crowd consists of providing agents with a broad library of motions to choose from, using behavioral rules and blending between the motions as the agents transition from one action to another. Behavioral selection of motion clips from a library should achieve good results if the library is large enough to provide satisfactory variation. However, this method is not always feasible for a small studio without access to an extensive library of motion clips or for a crowd animation with novel motion that cannot be motion captured. Furthermore, it does not provide an easy way to tune the crowd motion to fit a particular style.

We have developed an interactive evolutionary design system that will help designers create a diverse set of crowd motions that belong to a few families of expressive motion and can be tailored to fit the designer's preferences. Our system generates a range of motion variants using a single input motion. The input can be in the form of keyframe animation sequences and motion capture data should work as well. Therefore our method can not only be easily used to amplify a small database of motion clips for crowd animation but also as an aid in the ideation process in motion development for a character.

The system alters the input animation via user-defined free form deformations which reshape the animation channels specifying the motion. Attributes that control deformations act as genes within our design environment's genetic algorithm. The resulting population of motions can then be filtered using several techniques described in this paper that enforce a set of constraints. Physical constraints can be utilized to define the physical properties of each motion. The designer views these filtered motions at each generation to interactively determine the fitness of the crowd members. Through continued interaction, the designer can guide the population to one or more areas of the space of solutions which portray an interesting and desirable set of emotions or expressions. Interesting motions can be saved in a library for use later in the design process or as part of the final set of motions.

Our system also has the benefit of providing the designer with not just static samples from a space of possibilities, but also a neighborhood of similarity around each motion that can be used to tweak a single motion if desired. This flexibility is an improvement over a static library of motion capture clips in the same way that images produced by a procedural shader are more flexible than a set of scanned images. Motions generated by the system can be used to inspire animators with new movement ideas or stored in a database for generating crowd animation.

In the interest of generalization in the area of time-varying designs, we have extended our system to design particle system motion as well. We have shown that the same set of techniques that allows users to design character motion variants also works with particle systems. Thus the system could be used to explore variants on a particular special effect as well as for designing a set of particle systems, such as a group of clouds, that move in similar, stylized ways.

2 Related Work

Our approach draws from previous work in areas ranging from genetic algorithms to motion variation techniques and from crowd design methods to particle system evolution.

2.1 Evolutionary Design

Evolutionary design makes use of either genetic programming or genetic algorithms to search a space of possible designs in a given model and provide a solution that satisfies a particular fitness function. Sims paved the way for the use of evolutionary design in animation with a novel system for creature modeling and control [1]. Gritz and Hahn introduced a genetic programming method for evolving optimal motion for simple articulated figure motion [2]. Both of these methods used offline calculations to produce the resulting animations.

Interactive evolutionary design, unlike many genetic algorithm applications, lets the user interactively determine the fitness of the evolved solutions. Sims applied this technique to a wide array of application areas, including procedural models and textures [3].

In the application area of character animation Lim and Thalmann presented an intuitive interface for searching through a design space by selecting one from a pair of options in “tournament” style [4]. The same authors also presented a method for interactively evolving single walk cycle animations, using a genetic algorithm to scale and offset joint angle trajectories [5]. Ventrella introduced a physically-based animation and creature design system that allows users to interactively determine fitness of individuals explicitly by altering fitness functions as well as implicitly by selecting individuals of high fitness. His motion model consisted of single DOF joints activated by parameterized sinusoidal functions for the sake of fast interactivity [6]. We extend his novel work in this area by introducing a flexible framework for generating motion which allows for a wide range of expressive motion. Marks et al. describe a novel interface for parameter-setting problems with emphasis on dispersion and arrangement for optimal visual browsing of the search space [7]. For a more complete overview of this area of research, we refer the reader to Lewis who has provided an extensive survey covering a wide array of techniques and applications for evolutionary design [8].

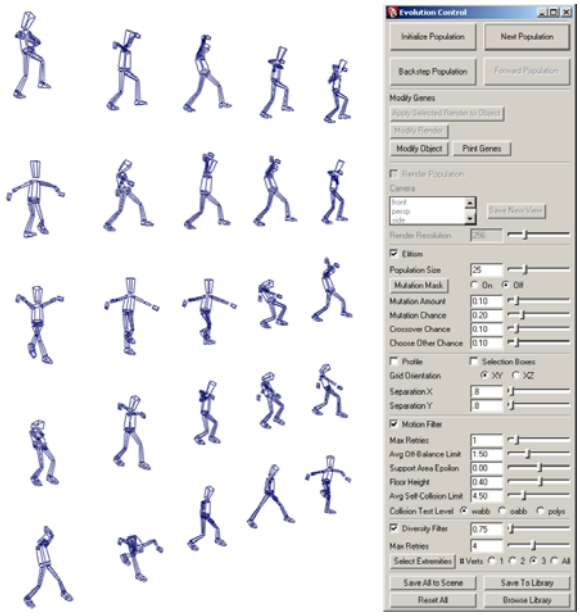
2.2 Motion Variation

There are a wide variety of ways to generate motion variation. Amaya et al. use signal processing techniques to embed varying emotions into a neutral motion clip [9]. Sung proposes a method for synthesizing motion clips for crowd animation using motion graphs [10]. Chi et al. introduce a system for modifying human motion using a Laban inspired effort and shape parameterization [11]. Neff and Fiume introduce a method that uses both low-level and composite properties to edit character motion iteratively and interactively [12]. Wang et al. present a motion signal filter method for making a motion more animated or cartoon-like [13]. Gleicher has provided an extensive survey of constraint-based motion editing techniques with particular attention paid to “per-frame inverse kinematics plus filtering” techniques [14].

2.3 Crowd Design

Crowd design has typically centered on the problems of navigation and behavioral patterns. For example, Kwon et al. present a graph-based approach for intelligently deforming group motion trajectories [15]. In addition, Treuille et al. introduce a particle-based solution for crowd navigation without the use of agent-based dynamics [16]. Li and Wang have used interactive evolutionary design to tune the parameters in a virtual force based system [17]. Sung et al. presented an efficient statistics-based scalable model for goal-directed crowd behavior that can satisfy duration, orientation, position, and pose constraints for individuals in the crowd [18]. Musse and Thalmann created a real-time interactive system with three methods for editing crowd behaviors [19]. Similarly, the crowd simulation middleware in use today focuses mainly on navigation and behavioral rules while relying on extensive motion libraries to provide variation of motion [20]. Time warping and blending between actions selected by behavior models are a popular means of generating motion variation in crowds.

Our method seeks to complement prior work in this area by addressing the need that crowd designers may have to craft the expressive motion of crowd members. We



Render Population
Camera: front, persp, side
Render Resolution 256
[✓] Elitism
Population Size 25
Mutation Mask: On [✓ Off]
Mutation Amount 0.10
Mutation Chance 0.20
Crossover Chance 0.10
Choose Other Chance 0.10
Profile
Selection Boxes
Grid Orientation [✓ XY] XZ
Separation X .8
Separation Y .8
[✓] Motion Filter
Max Retries 1
Avg Off-Balance Limit 1.50
Support Area Epsilon 0.00
Floor Height 0.40
Avg Self-Collision Limit 4.50
Collision Test Level [✓ waabb] oabb polys
[✓] Diversity Filter 0.75
Max Retries 4
Verts 1 2 [✓ 3] All

Fig. 1. Interactive evolutionary design interface - the phenotypes are arranged in a grid for easy viewing and selection with the mouse

choose to focus exclusively on generating the variety of expressive motion portrayed by the crowd. We encourage use of our system in conjunction with the existing methods for navigation and behavior.

2.4 Particle System Design

Evolutionary techniques have been applied to the creation of particle systems, starting with the paper by Sims which used CA lookup tables as the genotype representation [21]. Hastings et al. describe a system for the interactive evolution of particle systems using artificial neural networks as the base representation for each particle system [22,23].

3 Interactive Evolutionary Design

There are often two types of designers involved in evolutionary design: the meta-designer and the designer. The meta-designer defines the search space by creating the original prototype to be altered by the genetic algorithm. In our case this is a motion clip with parameterized deformers applied to it. The meta-designer also defines which attributes are adjustable and gives ranges to these attribute values. There must be enough meaningful attributes to provide a sufficiently rich search space for the algorithm to traverse. The designer who directs the evolution process will be able to explore solutions that exist in the space defined by the meta-designer.

Understanding the mechanics of the genetic algorithm supporting the search process can yield faster convergence rates. For example, more experienced designers who understand the inner workings of the reproduction algorithm may choose an individual based on a specific trait that they recognize as connected with a gene value that is desirable and important, even though a global perceptual evaluation may not have compelled the designer to choose that individual due to the influence of other undesirable genes. In subsequent generations that specific trait will likely show up again, perhaps accompanied by different, more desirable traits, yielding a higher fitness population more quickly.

The technique described in this paper assumes that a parametric model with a rich search space has already been designed. Problems in the field of parametric modeling are both challenging and interesting, but they are beyond the scope of this paper. Furthermore, the solutions to parametric modelling problems are typically domain specific. Bezirtzis, et al. have given attention to some of the general concerns of parametric search space design in their industrial design case study [24] with emphasis on the fact that the design of a parametric space is an iterative process that can only be verified empirically. In general, the areas of high fitness in a parametric search space for interactive evolutionary design should be much larger than those in typical non-interactive evolutionary design applications. This is due to the small number of individuals that can be viewed in one population by the human visual system and the low number of generations required for convergence in a reasonable interactive environment. Also, discontinuities or sharp features in the search space should be avoided as much as possible. Otherwise, the designer will find that slight changes in gene values can result in dramatically different phenotypic expression.

Our evolutionary design method takes advantage of the extraordinary perceptual abilities of the human visual system by showing the designer entire generations of motion all at once (See figure 1). This data presentation style, called “small multiples” by Tufte [25], not only provides a better visual comparison of the crowd motions but also gives the designer a better idea of the cumulative crowd motion during the design process than if the designer had to evolve and evaluate motions individually.

In our system the list of attributes belonging to the parametric model corresponds to a genotype which is represented by a fixed-length array of floating point numbers (gene values). The phenotype in our system is the motion produced when these gene values are mapped to the model’s parameters. The designer chooses a set of parents that will participate in the reproduction process for the next generation. Our genetic algorithm chooses two distinct, random parents from this set each time it produces a new offspring. It copies the genes from one parent and then switches to copying the other parent’s genes with a user-defined crossover probability. The genes are then mutated given a user-defined probability by adding a random value between -1 and 1 scaled by a user-defined mutation amount.

There are many variants on reproduction algorithms for genetic search, and we are not bound to this particular algorithm. However, the ability to let the user choose more than just two parents for the next generation provides a very nice property for the evolutionary design of crowd motions: The designer is able to simultaneously engineer distinct subspaces of motion for an entire crowd. Alternately, the designer can save interesting motions from exploration of one area of the search space into a motion library and can reintroduce these motions later, when exploring other areas of the search space. Phenotypes can be saved to the library as hardware rendered videos of the actual animation or of a revolving view of the phenotypic form at a particular frame. In addition, we encourage designers to explore different areas of the space by using a functionality in our interface that allows designers to backstep to previous generations and explore characteristics found in previously ignored parts of the population.

It is important to note that our system cannot provide enough variation from one input motion to create all the types of action that might be required in a crowd scene. For example, it cannot turn a walking motion into a sitting motion or vice versa. Instead it provides varieties of walking motions that can be placed in a motion clip library along with varieties of sitting motions that were generated from a separate input motion. Also, an animator cannot design just any particular preconceived motion using our system. It will only be possible to generate motions that exist in the available search space created by the meta-designer. Furthermore, if an animation designer has a specific visual expressive quality in mind, it can often be animated more quickly using conventional methods. This system is more useful for discovering novel motions and expressive qualities during exploration of the search space. Interactive evolutionary design has typically been aimed at aiding designers in the ideation phase of development of a character or idea, and the same applies here. Our system can be used to craft a novel motion for an individual character or a set of characters, but it may also serve as inspiration in the planning stages of new character motion development.

4 Motion Generation

Our method requires interaction from the designer at each iteration of the algorithm (see figure 2). We begin by preparing the input motion for modification by the genetic algorithm which, at the discretion of the designer, sends the resulting motion through a number of constraint-based filters and a dispersion algorithm.

4.1 Input

The underlying representation of our motion evolution system consists of the set of animation channels taken from the input motion clip. We do not use a simplified motion model based on sinusoidal signals or any other similar technique. Therefore our system is not limited to cyclical motions, but can be used to generate variants on any type of motion that can be keyframed. Each animation channel is converted to piecewise Bezier spline geometry based on the data frames and their tangents. The meta-designer sets up deformers [26] to alter these splines and also defines which attributes of the deformers can be changed by gene values from the genetic algorithm. The splines are then altered by these deformers during the evolutionary design process. Once the spline geometry has been changed, the information is then transferred back to the animation channels of the individual being evolved. The resulting motion is displayed to the designer. We set the system up this way so that any 2D deformation that does not violate the one-to-one correspondence of the function represented by the animation channels may be applied,

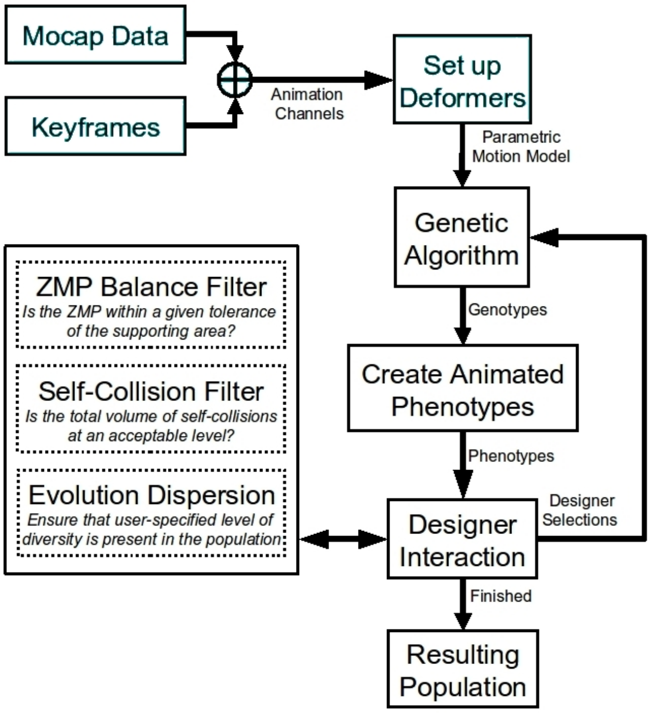


Fig. 2. Pipeline Overview

giving the meta-designer complete flexibility in creating and controlling the space of possible motions.

See figure 3 for an example of how lattice deformers can be applied to an animation channel. The first curve (3a) and accompanying animation sketch represent the original motion. Notice the lattice is in its initial, undeformed state. The second curve (3b) demonstrates the effect of translating and scaling the inner lattice points along the X-axis. The original curve and lattice are shown in the background for comparison. The resulting animation shows a faster backward swing of the forearm and a slower forward swing. The third curve (3c) demonstrates the subtle results of reducing the rotation values during the middle section of the animation sequence. The fourth curve (3d) shows the effect of increasing the rotation values at the beginning and end of the arm swing. The same lattice defomer and animation channel technique transfers quite well to particle systems where certain qualities, such as position, color, force, mass, and velocity can be keyframed over time. Particle system characteristics that are not keyframed can also be evolved by simply allowing the genetic algorithm to alter the constants that control those traits.

Ideally, the parametric models produced via the application of lattice deformers will provide a wide range of variation with few problems. However, some parameters may fight each other and occasionally lead to unrealistic or undesirable results. We address this by adding a layer of filters to the pipeline that automatically detect problematic

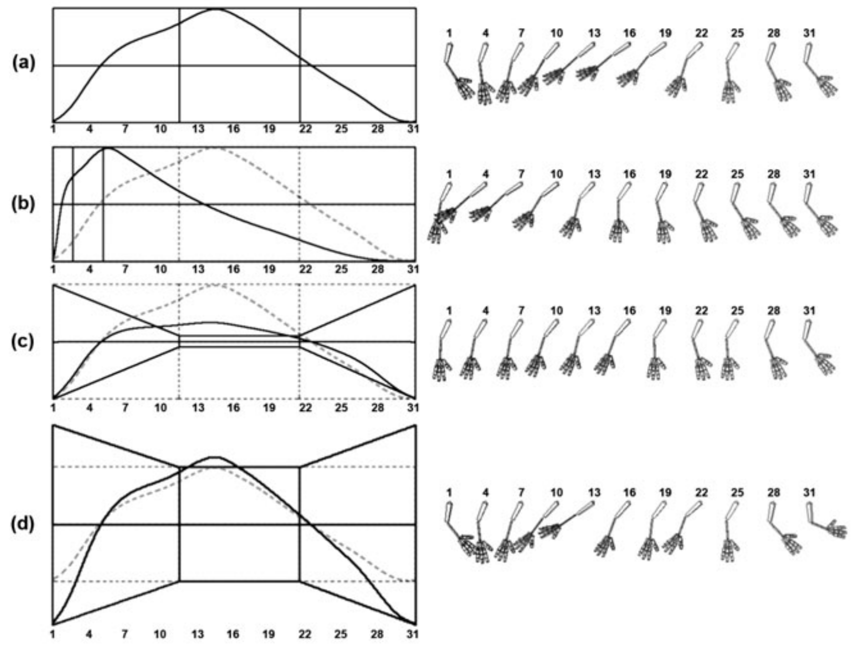


Fig. 3. A variety of motions achieved using free form deformations on the rotation channels for the arm. Only the elbow joint values are represented by the graphs, but, in the interest of maintaining proper timing, the same deformations are being applied to the shoulder joint in the animation sequence to the right. The X-axis of each lattice controls the passage of time (labeled in frames), and the Y-axis values control the rotation angle of the joint at any point in time.

results above a tolerance threshold and generate individuals to replace these unacceptable phenotypes. Though it would be ideal to correct these phenotypes, we choose to replace them in the interest of faster interactivity. This will be discussed further below. These filters, though general in some regards, should specifically apply to the type of motion being generated. In the case of crowd character motion, any number of high-level qualities of locomotive animation can be addressed here, but we choose to focus on two particular properties of motion where we have observed the abuses of large variation: balance and self-collision.

4.2 Balance

In order to measure the level of balance in individual phenotypes and decide if they are acceptable we have adopted the zero moment point (ZMP) algorithm of Tak et al. ZMP in dynamic motion analysis is similar to the center of gravity in the static case. It is defined as the point on the ground plane under a character where there is zero moment. In other words, if this point were modeled as a joint between the character and the ground, there would be no actuation at this joint. As a character moves, the ZMP will create a trajectory over the ground plane.

Our balance filter ensures that the ZMP is always within the character's support area. The support area is the convex hull of the contact area between the feet and the floor. This definition encapsulates both the single and double stance phases of bipedal motion and allows for seamless calculation between the two phases. In some cases, the designer may want the motion of individuals in the crowd to appear exaggerated or cartoon-like. In a case such as this, a precise balance constraint might cause more harm than good. Therefore we have implemented the filter in such a way that the designer has control over the allowable balance error. We define balance error as the distance from the ZMP to the closest point on the support area, and we sum this error over the entire motion sequence.

$$\sum_{i=1}^{F} \|ZMP_i - closestPoint(ZMP_i, K \cdot SA_i)\| \tag{1}$$

F is the number of frames in the animation, ZMP contains the zero moment point trajectory over all the frames, and SA contains the support area hull over all the frames. We also allow the user to enforce a lesser or greater degree of balance on the entire sequence by adding a custom support area scale factor K. The value of K can be adjusted through the system's interface thus allowing the user to shrink the support area to constrain the ZMP error more towards the center of support if more stable motion is required.

4.3 Self-collision

Since the variation of arbitrary animation channel attributes may introduce self-collisions, we have developed a set of tests to filter out these self-colliding motions. We employ a simple bounding box method, testing for overlap of world-aligned bounding boxes and then calculate the volume of object aligned bounding boxes if two links are found to be intersecting. We then sum the volumes of all the intersections.

$$\sum_{i=1}^{F}\sum_{j=1}^{L}\sum_{k=j+1}^{L}\{volume(O_j \cap O_k) \mid A_{j,k} = 0,\ W_j \cap W_k \neq \emptyset\} \tag{2}$$

F is the number of frames in the animation, L is the number of links or bones in the character, O contains the object-aligned bounding boxes of the links, W contains the world-aligned bounding boxes, and A is an adjacency matrix which describes the spatial adjacency of the links. By summing the total volume we avoid over-penalizing quick, grazing collisions while still penalizing quick, high-volume collisions as well as slow, grazing collisions. We ignore collisions that are design artifacts of the given model's geometry as well as collisions between adjacent links. Since we do not test at the polygon-polygon level for intersection in order to reduce running time, this filter only serves as a heuristic and not an exact measure of self-collision. Nevertheless, in practice, it gives a good indication of the level of self-collision for a given phenotype, especially since we want to keep computation to a minimum in the interest of higher levels of interaction.

4.4 Replacement Method

It would be preferable to fix unacceptable phenotypes and bring them back into the allowable space of motion defined by our balance and self-collision constraints. In fact, the method of Tak et al. optimally transforms unbalanced motions into balanced ones [27]. There are also methods for correcting self-colliding animation via inverse kinematics optimization [28]. However, we do not attempt to fix unacceptable motions because this would require an optimization process. Optimization would introduce longer wait times between generations, reducing the interactivity of our evolutionary design environment. Instead, we generate a new individual motion to replace the old one and resubmit it to the filters for evaluation. We are able to proceed in this way because our space of possible motions was created from a balanced original motion that was free from self-collisions and so the rich search space of variants will provide a viable, balanced replacement motion within a few iterations. Furthermore, the reproduction process uses the user-selected motions as input for any newly generated replacement motion so the offspring will most likely have similar balance and self-collision error to the parents' errors. Even so, generating replacement individuals with our reproduction algorithm may not always result in an acceptable individual, especially if the user's constraints are too restrictive or if the search space is not rich enough. If this occurs, we let the replacement process run for a user-defined maximum number of iterations and then force it to move on, replacing the unbalanced individual with the most balanced option found so far.

4.5 Diversity

Although duplicate motions in a crowd are harder to spot than duplicate appearances, they can be spotted just as easily whether or not each character has a different appearance. Moreover, varied motion between two visually equivalent individuals can help to obscure the fact that they have a similar appearance [29]. It follows that diversity of motion provides desirable qualities for crowd animation. Genetic algorithms by their very nature seek convergence to a specific area of the search space. Because we want to design coherent sets of motion and because proximity in the search space corresponds to phenotypic similarity in our system, this convergence is beneficial. However, it can also be problematic since we want to find a diverse group of motions. We do not want the algorithm to converge so far that the resulting population becomes homogenous.

The designer in the interactive evolutionary design paradigm exercises ultimate control over how far the population converges. As mentioned earlier, if the designer chooses a variety of individuals as parents in one generation, the chances of diversity in the next generation are greater. However, in order to help the designer encourage diversity in the population and to ensure sufficient sampling of the local search space surrounding the designer's regions of interest, we run the evolution dispersion algorithm introduced by Marks et al. [7] at each generation. This algorithm alters the genes of individuals in a way that will discourage similarity between phenotypes. We measure the phenotypic difference between two individuals by comparing their point clouds. Our method is somewhat similar to the technique used by Kovar, et al. in their paper on motion graphs [30]. The point cloud for a character consists of the positions of a subset of the character's vertices over time. We consider the distance between point clouds to be the sum of the root mean squared error between all the corresponding vertices of two point clouds. We allow the designer to specify which parts of the model to choose vertices from when making this comparison. The designer can also specify the level of dispersion required at each generation.

Since this metric is based on point clouds, it transfers quite naturally to particle systems. The application of this metric is straightforward in the case of spatial qualities, and can be easily extended to include other qualities by increasing the dimensionality of the point cloud space.

4.6 Using the Filters

Because the motion filters require extra computation, using them will inevitably result in longer wait times between generations. The balance and self-collision filters are most useful during the first couple generations of the design process when the algorithm is sampling a wide area of the search space. These filters make the designer's job easier by avoiding the areas of the search space where two or more parameters fight each other and produce unwanted motion artifacts. As the design process progresses and the algorithm begins to converge, these filters should be turned off to speed up the interactivity of the system. In contrast, the diversity filter should only be used near the end of the design process when the algorithm is sampling a smaller area of the search space. In the case of particle system evolution, one would only want to use the diversity filter, as zero moment point and self-collision do not apply to massless particles. Using the above filters in this way will maximize the fitness of the options presented to the designer throughout the process in a way that is customizable to fit the designer's needs.

5 Results

We implemented our interactive evolutionary design motion generation method in a layer of MEL and Python code over Autodesk's Maya environment. We chose to use Maya because it is flexible enough to apply evolutionary design not only to animation, but also to modeling, texturing, and special effects domains. We also chose this software environment because most of the designers at our research facility are familiar with its interface, and we would like them to be able to create and evolve designs from their own parametric models.

Table 1. Average time (sec) and average number of rejected phenotypes while producing a generation of size 25

Filter	None	Balance	Collide	Disperse
Time	10.0	76.0	47.8	28.0
Reject	-	1.5	3.5	2

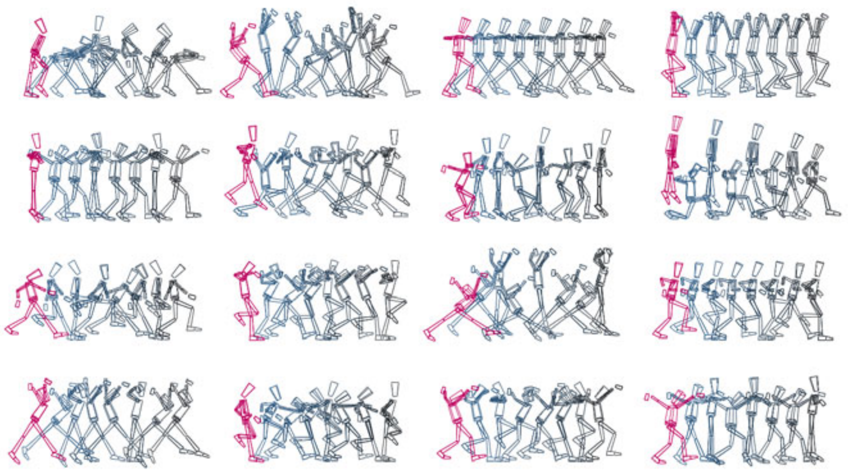


Fig. 4. A variety of sixteen motions evolved from a single walk cycle motion clip

Our hardware consists of an Intel Xeon 2.66 GHz cpu with 4 GB of RAM and an Nvidia Quadro FX 5600 graphics card. Performance metrics of our current implementation under varying conditions can be seen in Table 1. Each filter was set up with a maximum of 2 replacements per individual except for the dispersion filter which had a maximum of two replacements per population.The number of replacements required at each generation depends on the richness of the search space as well as on the designer's selections and the constraint thresholds set by the designer. The time required to replace an individual that does not meet the constraints is approximately equal to the time required to generate the original individual. The running time is dependent on the size of the population so smaller populations will take less time per individual.

For these tests, we created a parametric walk cycle model with lattice deformers on 14 different animation channels. Note, however, that any type of deformer available in Maya may be utilized. The deformers' transformations were controlled by 14 floating point numbers from the fixed-length array that formed the genotype of each individual. Although the size of the population is adjustable by the user, we generally use populations of size 25 because larger populations are harder for the human visual system to fully and easily comprehend.

Figure 4 shows a sampling of the variety of motions that can be produced from a single parameterized walk cycle. These motions are shown in an animation sketchbook style where every fourth frame is drawn to show the change in form over time. We are displaying the motions in animation sketchbook style only for the purposes of the paper. In our evolutionary design interface, the motions are displayed as time-varying animation.

Figure 5 shows a variety of particle systems in a sample initial population. In this case, the particles leave trails of spawned particles behind them so you can visualize the movement of the system even in a static image.

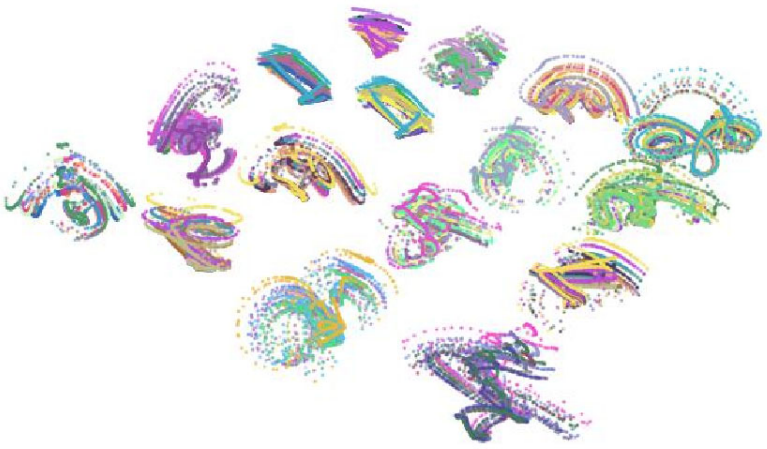


Fig. 5. A variety of sixteen particle systems evolved from a input system

We have conducted a user study to determine if our software truly provides an intuitive way for designers to create varieties of motion. In this study we asked three graduate students from the department of design who were already familiar with the Maya software environment to participate. After a brief 10-15 minute tutorial on how to use the system, each designer explored the space of options provided by the deformers in our parametric model of a walk cycle motion and decided on a style of motion to interactively develop. The entire process, including the tutorial, search space exploration, and motion design took anywhere from an hour and a half to two hours for each designer. The designers all felt that the system was easy to learn and enjoyed experiencing the interactive evolutionary design process. They agreed that the motions produced would make for interesting, varied background crowd character motions or as a fertile starting point for the development of a character motion with an individualized style. However, they said that the motion produced would need to be refined with a high-level of control over specific keyframes if it were to be used for foreground or hero characters. The populations they created (See the animations on the project web site at http://accad.osu.edu/Projects/Evo/) including a set of energetic dancers, a group of feminine characters, and a mob of zombies demonstrate the wide variety of motions that are acheivable from just one input motion.

6 Future Work

There are many more ways to filter the individual motions that could prove to be helpful to the crowd designer. A few such filters that could be useful include: joint torque analysis based on limits from a comfort model [31], various methods of psychological analysis (aggressiveness, energy level, coordination, etc.), Laban movement analysis (effort and shape), as well as gender or age analysis. The balance and self-collision filtering could be run in parallel on subsets of the population using multi-core processors to speed up computation times and enhance the interactivity of the system.

7 Conclusions

Our method presents a novel approach to evolving families of expressive motion, making it easier for a crowd designer to quickly and intuitively find a satisfying combination of motion variations for a specific application. This method could prove especially useful to those who do not have access to motion capture facilities or cannot afford to spend time capturing a wide range of motion clips. Our interaction model allows the user to view and make decisions about entire generations at once, and our reproduction algorithm allows for evolution of multiple (even mutually exclusive) styles of motion simultaneously. Our use of user-defined constraints plus the designer's selections as the determination of fitness exemplifies a hybrid system that seeks to maximize the designer's time and attention in the evaluation of populations by filtering out the individuals who do not meet the given criteria.

References

1. Sims, K.: Evolving 3d morphology and behavior by competition. Artificial Life 1(4), 353–372 (1994)
2. Gritz, L., Hahn, J.K.: Genetic programming for articulated figure motion. Journal of Visualization and Computer Animation 6, 129–142 (1995)
3. Sims, K.: Interactive evolution of equations for procedural models. The Visual Computer 9(8), 466–476 (1993)
4. Lim, I.S., Thalmann, D.: Tournament selection for browsing temporal signals. In: Proceedings of Symposium on Applied Computing 2000, pp. 570–573. ACM, New York (2000)
5. Lim, I.S., Thalmann, D.: Pro-actively interactive evolution for computer animation. In: Proceedings of Computer Animation and Simulation 1999, pp. 45–52 (1999)
6. Ventrella, J.: Disney meets darwin-the evolution of funny animated figures. Computer Animation (1995)
7. Marks, J., et al.: Design galleries: a general approach to setting parameters for computer graphics and animation. In: Proceedings of SIGGRAPH 1997, pp. 389–400. ACM Press/Addison-Wesley Publishing Co., New York (1997)
8. Lewis, M.: Evolutionary visual art and design. In: Romero, J., Machado, P. (eds.) The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music, pp. 3–37. Springer, Heidelberg (2007)
9. Amaya, K., Bruderlin, A., Calvert, T.: Emotion from motion. In: Graphics Interface 1996, pp. 222–229 (1996)
10. Sung, M.: Continuous motion graph for crowd simulation. In: Hui, K.-c., Pan, Z., Chung, R.C.-k., Wang, C.C.L., Jin, X., Göbel, S., Li, E.C.-L. (eds.) EDUTAINMENT 2007. LNCS, vol. 4469, pp. 202–213. Springer, Heidelberg (2007)
11. Chi, D., Costa, M., Zhao, L., Badler, N.: The emote model for effort and shape. In: Proceedings of SIGGRAPH 2000, pp. 173–182. ACM Press/Addison-Wesley Publishing Co., New York (2000)
12. Neff, M., Fiume, E.: Aer: aesthetic exploration and refinement for expressive character animation. In: Proceedings of SCA 2005, pp. 161–170. ACM Press, New York (2005)
13. Wang, J., Drucker, S.M., Agrawala, M., Cohen, M.F.: The cartoon animation filter. ACM Transactions on Graphics 25(3), 1169–1173 (2006)
14. Gleicher, M.: Comparing constraint-based motion editing methods. Graphical Models 63(2), 107–134 (2001)

15. Kwon, T., Lee, K.H., Lee, J., Takahashi, S.: Group motion editing. In: Proceedings of SIGGRAPH 2008, pp. 1–8. ACM, New York (2008)
16. Treuille, A., Cooper, S., Popovic, Z.: Continuum crowds. ACM Transactions on Graphics 25(3), 1160–1168 (2006)
17. Li, T.Y., Wang, C.C.: An evolutionary approach to crowd simulation. Autonomous Robots and Agents, 119–126 (2007)
18. Sung, M., Kovar, L., Gleicher, M.: Fast and accurate goal-directed motion synthesis for crowds. In: Proceedings of Symposium on Computer Animation 2005, pp. 291–300. ACM Press, New York (2005)
19. Musse, S.R., Thalmann, D.: Hierarchical model for real time simulation of virtual human crowds. IEEE Transactions on Visualization and Computer Graphics 7(2), 152–164 (2001)
20. Massive Software: Massive prime (2009), www.massivesoftware.com/prime
21. Sims, K.: Artificial evolution for computer graphics. In: Proceedings of SIGGRAPH 1991, vol. 25, pp. 319–328. ACM Press, New York (1991)
22. Hastings, E.J., Guha, R.K., Stanley, K.O.: Neat particles: Design, representation, and animation of particle system effects. In: IEEE CIG 2007 (2007)
23. Hastings, E.J., Guha, R.K., Stanley, K.O.: Interactive evolution of particle systems for computer graphics and animation. Trans. Evol. Comp. 13(2), 418–432 (2009)
24. Bezirtzis, B.G., Lewis, M., Christeson, C.: Interactive evolution for industrial design. In: C&C 2007:Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition, pp. 183–192. ACM, New York (2007)
25. Tufte, E.R.: Envisioning Information. Graphics Press (May 1990)
26. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. In: Proceedings of SIGGRAPH 1986, vol. 20(4), pp. 151–160 (1986)
27. Tak, S., Song, O.Y., Ko, H.S.: Spacetime sweeping: An interactive dynamic constraints solver. In: Proceedings of Computer Animation 2002, pp. 261–271. IEEE Computer Society, Washington, DC (2002)
28. Müller, A.: Collision avoiding continuation method for the inverse kinematics of redundant manipulators. In: Proceedings of Robotics and Automation 2004, vol. 2, pp. 1593–1598 (2004)
29. Mcdonnell, R., Larkin, M., Dobbyn, S., Collins, S., O'Sullivan, C.: Clone attack! perception of crowd variety. In: Proceedings of SIGGRAPH 2008, vol. 27, pp. 1–8. ACM Press, New York (2008)
30. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. In: Proceedings of SIGGRAPH 2002, vol. 21, pp. 473–482. ACM Press, New York (2002)
31. Ko, H., Badler, N.I.: Animating human locomotion with inverse dynamics. IEEE Computer Graphics and Applications 16(2), 50–59 (1996)

Dual Phase Evolution as a Framework for Understanding Complex Adaptive Systems

Greg Paperin and Suzanne Sadedin

Clayton School of Information Technology, Monash University, Vic. 3800, Australia
greg@paperin.org, suzanne.sadedin@gmail.com

Abstract. Evidence from several fields suggests that dual phase evolution (DPE) may contribute to distinctive features associated with complex adaptive systems. Here, we review empirical and theoretical evidence for DPE in natural systems and discuss the relationship of DPE to self-organised criticality and adaptive cycles. We describe a general model for DPE in networks, and present preliminary data illustrating the emergence of phase changes.

Keywords: Dual phase evolution, Networks, Connectivity, Phase changes, Self-organised criticality, Adaptive cycle.

1 Introduction

Complex adaptive systems share several interesting properties such as self-organisation, far-from-equilibrium dynamics, perpetual novelty and sustained diversity. While many advances have been made in understanding specific complex adaptive systems (CAS), a unifying theory of their underlying mechanisms remains elusive. Several conceptual frameworks have been proposed to describe the properties of CAS. These include the concepts of self-organised criticality (SOC) [1, 2] and the adaptive cycle [3]. While these frameworks clearly capture some of the dynamics seen in CAS, other properties remain neglected and the causal processes have not been clearly defined.

Past research shows that CAS can be described in terms of networks of interacting components [4] and that structural properties of these underlying networks may explain many of the processes observed in CAS. Based on this realisation, the notion of Dual Phase Evolution (DPE) was proposed [5, 6]. DPE explains CAS properties such as perpetual novelty and diversity, modularity, and complexity on all scales as consequences of recurring phase transitions in connectivity and interaction patterns of underlying networks. DPE processes are observed across a wide range of CAS of various orders of magnitude: from species evolution and ecosystem development, to socio-economic systems, to artificial adaptive and optimisation systems.

Here, we review some of the empirical evidence for DPE and contrast it with established frameworks for understanding CAS dynamics, in particular SOC and the adaptive cycle. We highlight the key differences between these frameworks and DPE and discuss how some processes may be explained in terms of these different frameworks. This presents a step towards developing a holistic understanding of CAS dynamics based on underlying network properties. To support our arguments we briefly describe a simulation model of energy flow through a network of interacting components. Several real world CAS can be mapped to this network model. While a thorough analysis of the model dynamics is on-going, the results indicate that DPE processes emerge in the model under a wide range of parameters.

2 Dual Phase Evolution

2.1 Examples

Evidence from several fields suggests that phase changes in landscape connectivity form a powerful agent of evolutionary change and innovation. Disasters often mediate long-term changes in the composition of ecological communities, with established species forming an impenetrable barrier to invasion by novel species until massive population destruction clears the landscape. Palynological data show that changes in species composition in North American forests are consistently associated with major wildfires [7]. At larger geological timescales, many recent adaptive radiation events are associated with transitions between glacial and interglacial periods that lead to drastic changes in habitat connectivity for a wide variety of species [8]. Climate-change mediated variations in sea level can cause populations living at specific depths to become fragmented or connected, while temperature and rainfall variation alters the connectivity of lakes and waterways and their ecological communities [9]. For example, diverse new species of cichlids emerged in African rift lakes after the last Ice Age isolated local fish populations. Genetic suture zones (areas where locally differentiated populations meet) in many European and North American species including trees, insects, birds and mammals can be traced to population expansion from refugia that were isolated during glacial periods [10, 11]. Repeatedly-isolated refugia are associated with speciation events; for example, a meta-analysis of mitochondrial DNA studies in 63 bird species, showed that many adaptive radiations initiated in the Pliocene were completed when glaciers fragmented populations in the Pleistocene [12]. On the mountainous island of Sulawesi, adjacent-living similar species of grasshoppers, macaques, pond-skaters, cicadas, bees, butterflies and beetles are thought to have arisen during periods of habitat fragmentation caused by climate change [13].

At even larger scales, state transitions may be seen in evolutionary dynamics after environmental change. Eldredge and Gould [14] documented evidence for punctuated equilibria in the fossil record, arguing that biological history is dominated by long periods of stasis with occasional bursts of innovation after mass extinction. These bursts of innovation, according to Gould [15], are triggered by the removal of ecological specialists, opening up niches for exploitation by the widespread generalists which preferentially survive mass extinction. These generalists then undergo adaptive radiation. In this sense, evolution alternates between long, slow periods of general stability dominated by species selection (stability phase) and brief periods of rapid microevolution where novel adaptations arise (variation phase). There are several possible explanations for punctuated equilibrium [16]. However, the strong geological association between disasters (such as asteroid strikes, vulcanism and climate change), mass extinction and subsequent radiation events suggest that these external drivers are crucial in that they force the switch from stability to variation phases by altering the connectivity of food webs and landscapes.

Simulation experiments further support this argument. For instance, Paperin et al. present a model [17] in which organisms normally exist within a connected landscape in which selection maintains them in a stable state. Intermittent disturbances (such as fires, commentary impacts) flip the system into a disconnected phase, in which populations become fragmented, freeing up areas of empty space in which selection pressure lessens and genetic variation predominates. The simulation results show that DPE-like connectivity phase changes can facilitate the appearance of complex diversity in a landscape ecosystem.

Dual phase processes also occur in non-living natural complex systems. For instance, Perkins [18] describes in an overview article how various kinds of landscape patterns may have been formed by repeated phase changes in several interacting geomorphic processes. A well studied example of such landscapes – the geometric shapes of stones occurring in many polar and high alpine environments – has been investigated by Kesser and Werner [19] who demonstrated that such patterns may emerge through freeze-thaw cycles that drive an interaction between two feedback processes. In the first process, ice forms in freezing soil, segregating stones and soil by shifting soil toward soil-rich areas and stones toward stone-rich areas. In the second process, stones are transported along the borders of stone-rich domains, which are squeezed and shaped under the pressure of expanding freezing soil. The authors provide a numerical simulation model [19] that can reproduce the patterns found in natural landscapes of this kind [18].

Connectivity phase changes are also the driving force in many artificial CAS. Phase transitions of interaction networks have been implicitly present in many traditional optimisation algorithms in the form of mediation between local and global search. For instance, in simulated annealing [20, 21] the temperature schedule is used to arbitrate between local and global search steps. Similar ideas have been employed to improve performance in a variety of optimisation techniques that are prone to being caught in undesirable local optima when applied to non-smooth search spaces. This includes, for instance, the back propagation learning algorithm for artificial neural networks. [e.g. 22], the Particle Swarm Optimisation algorithm [e.g. 23, 24], Genetic Programming [e.g. 25] and Support Vector Machines [e.g. 26, 27]. In the above algorithms the connectivity of the transportation network along which the search proceeds is changed from well connected (global search, exploration) to poorly connected or disconnected (local search, exploitation).

In these artificial optimisation systems, phase transitions occur only once or a few times in one direction. However, natural DPE processes are typified by repeated connectivity phase transitions in both directions. Arguably, optimisation algorithms supplemented with simulated annealing style techniques may be improved by incorporating repeated connectivity phase transitions in both directions. An instance of this approach is a modification of the Cellular Genetic Algorithm [28, 29]. Kirley et al. [30, 31] modified this algorithm to supplement it with insights from population dynamics and landscape ecology. The evolving population was placed in a 2-dimensional cellular automation grid that is subjected to intermittent "disasters" that eliminate all solutions in one part of the grid. As a result, the population becomes fragmented and the gene flow between the sub-populations is diminished or interrupted. This allows the sub-populations to diverge and slows down convergence. Recombination of diverged sub-populations while re-populating areas freed by disasters often leads to discovery of new and fitter solutions. The Cellular Genetic Algorithm modified in this way outperforms the standard Cellular Genetic Algorithm on a number of hard test problems [30, 31].

It should be noted that in this case, the DPE phase transition occurs repeatedly in both directions. Two important interaction networks can be identified within the cellular grid. Firstly, there is the connectivity network between the populated grid cells. The connectivity of this network plays a role in determining the amount of gene (information) flow between different cells. Thus, connectivity in this network influences whether the population evolves as a whole or as divergent sub-populations. The second network is the connectivity network of free grid cells. These cells can be populated by newcomers without substantial competition. During phases where this network is well connected the algorithm has the opportunity to experiment with candidate solutions that may be less fit than some other part of the grid population, but that have potential to evolve towards a different, possibly better local optimum.

2.2 The DPE Framework

A common thread in all of the above examples is that complex properties of systems are mediated by qualitative changes in the connectivity structure of the underlying networks. The connectivity structure can be classified into two main states or phases: "connected" and "disconnected". The "connected" phase is typified by high edge density and short paths lengths. In this phase interactions can therefore occur between most of the network components. In the "disconnected" phase edge density is low, paths lengths are long, and the network typically consists of several disconnected components. Interactions in the disconnected phase typically occur locally or only within strongly connected components.

Since networks are inherent in the structure and behaviour of all complex systems [4], a connectivity avalanche [32] underlies many kinds of critical phase changes [33]. Therefore all such systems can switch between the two above phases. Systems in the disconnected phase tend to be balanced. They may exhibit strong local variability, but typically little large-scale variation. Global responses to external stimuli are constrained, as perturbations cannot propagate far. Systems in the connected phase, in contrast, exhibit less local variability, but significant variation on all scales in the sense that responses to external stimuli are generally hard to predict. The rich connectivity allows perturbations to propagate far, affecting many system parts [17].

DPE occurs when an evolving system repeatedly switches between these two phases (Fig. 1). Crucial for understanding many DPE systems is the mechanism responsible for these repeated phase transitions. There is much evidence that CAS generally self-organise towards a stable, balanced state. Stabilising forces include lower order dynamics, such as feedback loops, and higher order dynamics, such as selection (in a general sense) [34]. Analytical [35, 36] and computational [34] models show that lower-order local dynamics can stabilise systems over a large range of external forcing, and that higher order local dynamics (evolutionary dynamics) can greatly increase the stabilising effect. The adaptive forces that underlie global stability

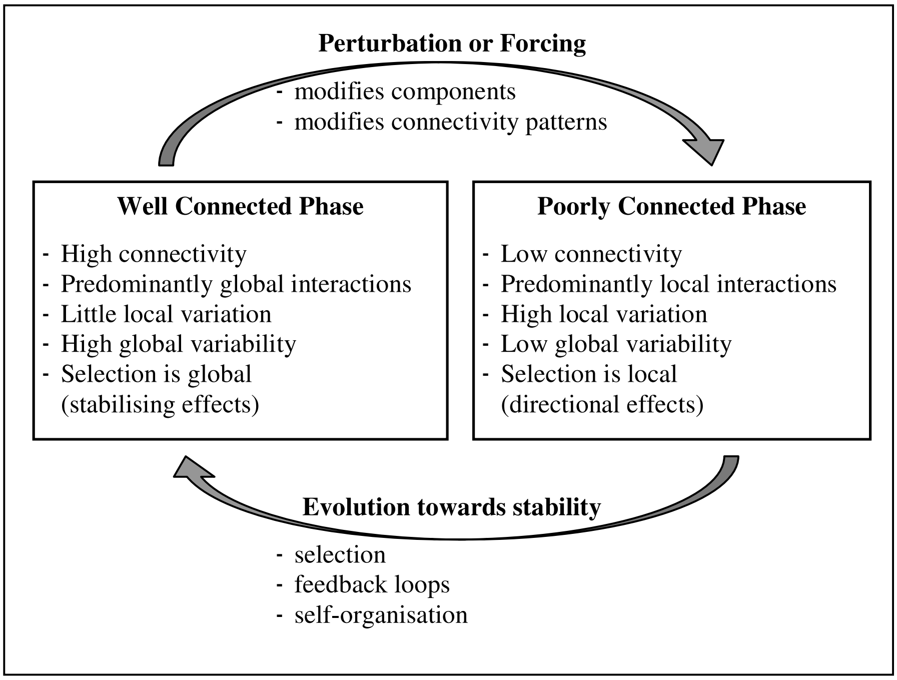


Fig. 1. The mechanism of Dual Phase Evolution. Systems flip between poorly connected and well connected phases. Perturbations or slow forcing – arising externally or internally - disrupt systems causing connectivity phase transitions in underlying networks. Internal pressures restore old and create new interactions.

of CAS also inhibit novelty and change. In particular, selection acting on system components at various scales, as well as on topology and interactions, may drive a system as a whole to a local optimum state, halting innovation [37]. Two mechanisms work against such long-term stasis.

One mechanism is co-evolution – a process here system components continuously adapt to each another in a feedback loop, thus providing some on-going innovation. However, co-evolution is not likely to account for the novelty observed in many CAS. For instance, current models suggest [38] that selection, not variation, drives biological speciation and that co-evolutionary feedback is likely to rapidly (on geological timescales) lead to stable local optima.

A second mechanism that may underlie continual novelty in CAS is disturbance. As discussed in section 2.1, evolutionary innovations often coincide with external perturbations. External disturbances may affect both system components and interaction networks, thus moving systems away from local optima. Densely connected interaction networks, while providing many stabilising interactions, also facilitate disturbance propagation. The complexity of dense interaction networks makes large-scale responses to disturbances essentially unpredictable.

Once away from a local optimum, systems enter a variation phase. Chance variation of local components may provide better adaptation to local constraints; selection facilitates proliferation of such changes within networks. Selection then amplifies variations and eliminates destabilising interactions, reducing connectivity, and components and their interactions self-organise towards new local optima.

Over time, surviving system components develop new interactions, increasing the connectivity of interaction networks that survived previous disturbances. Eventually, the system enters a new balance phase.

While some parts of a system may be completely or partly reorganised during a variation phase following a particular disturbance, others remain stable. These stable parts may form new interactions and assume new roles, acting as functional components during a variation phase. A simulation by Paperin et al. [39] demonstrated that DPE can result in modular networks. We conjecture that this mechanism may also contribute to emergence of hierarchical organisation in CAS.

2.3 DPE and Self-organised Criticality

DPE can be linked to several other key concepts in CAS theory. One such concept is Self-Organised Criticality (SOC) [1, 2]. Under SOC, CAS self-organise to a critical state where system behaviour emerges from propagation of stimuli via local component interactions. SOC suggests that CAS evolve towards the "edge-of-chaos" [40, 41], a transition state between the stasis of equilibrium systems and the unpredictability of chaotic systems.

Sizes of stimuli propagation avalanches in SOC systems follow a power distribution, leading some researchers to argue that power-distributed data imply SOC. Models [1] suggest ways in which certain natural systems may exhibit SOC dynamics. However, the general applicability of SOC remains doubtful. Other processes also lead to power-law distributed data. For example, it has been proposed [42] that the biosphere self-organises to a critical state, potentially explaining punctuated equilibria [14]. However, [43] demonstrates a non-critical extinction model that yields a power-law with an exponent closer [34] to the empirical punctuated equilibria data. SOC also appears to require fine-tuning of an order parameter [44, 45], and the applicability of SOC to non-conservative systems [44, 46] remains unclear.

To describe DPE using the SOC-vocabulary: CAS develop to a balance-state, where they are stabilised by internal forces (e.g. selection, negative feedback mechanisms). External disturbances repeatedly push a system *across the critical region*, to a chaotic state (in the sense that systems responses to stimuli are unpredictable), from which the system returns to a new balance-state, accumulating order and complexity on the way (Fig. 2).

Often, SOC is used to express that a system has self-organised to a specific state, without describing the underlying processes. The DPE framework attempts to define the internal forces responsible for system states. In this sense some systems may self-organise to a critical state through DPE. For instance, scale-free networks [47] are traditionally associated with SOC because their node degrees follow a power distribution. Traditionally, scale-free topologies were thought to arise through preferential node attachment during network growth [47]. However, scale-free

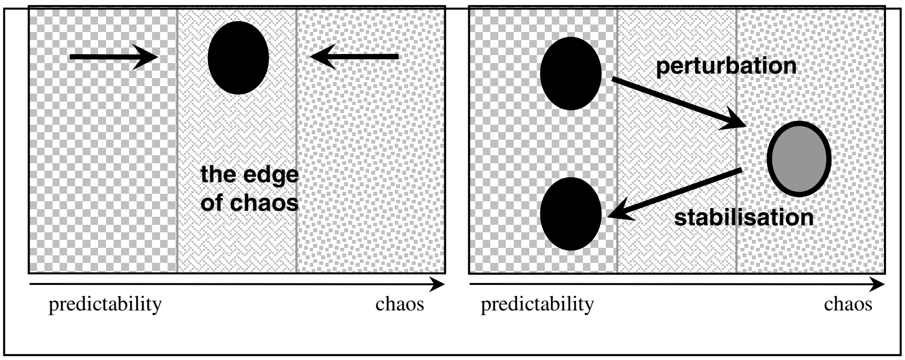


Fig. 2. Self-Organised Criticality vs. Dual Phase Evolution. SOC-theory suggests that CAS self-organise to a critical transition state between the general stasis of equilibrium systems and the random behaviour of chaotic systems (left). According to DPE, CAS are repeatedly pushed from a balance-phase (high connectivity) to a variation-phase (low connectivity) by external stimuli (right). The X-axis on this metaphoric illustration represents the degree of predictability of system's responses to stimuli.

topologies can arise through DPE in networks of constant size [39]. Networks developed this way may underlie some systems with apparent SOC dynamics.

2.4 DPE and the Adaptive Cycle

An influential concept in CAS theory is the adaptive cycle (AC) (see Gunderson and Holling [3]). The AC extends the idea of ecological succession [48], and is predominantly applied to ecological and socio-ecological systems, especially with reference to ecosystem management and resilience. The AC identifies 4 phases in ecological succession:

- a growth and exploitation phase (designated r), in which new or freed-up areas and niches are rapidly populated by opportunistic organisms;
- a conservation phase (K) signified by competition, selection and resource accumulation;
- a collapse or release phase (Ω), in which accumulated resources are catastrophically released, often mediated by disturbances;
- a reorganisation phase (α) in which the remains of an Ω-collapse are reorganised and restructured.

The AC concept attributes typical CAS properties to each phase. Resilience against external forcing is expected to be high during r and α phases but low during K, while resource availability is high during α and K phases, but low during r and Ω. Connectedness of control variables is maximal near the end of a K-phase. The AC provides a descriptive formalism for self-organisation in ecosystems. DPE theory distils concepts of the AC that are applicable to a wider range of CAS and provides a causal model based in network theory.

The balance phase in DPE loosely corresponds to the *r*-*K* transition in AC. This phase is signified by stabilising selection, increasing connectivity, and growing potential for disturbance propagation. The variation phase in DPE loosely corresponds to the Ω-α-*r* transition in AC. This is a phase of innovation and re-organisation of underlying networks.

Notably, connectedness in AC refers to the richness of interactions of control variables. In fact, there may be several interaction networks with different connectivity regimes within a system at any one time. For example, species in food webs and populations in landscapes form interaction networks that act simultaneously on the same groups but may have very different topologies. The structural properties of the interaction network of control variables may thus be different from the interaction network of components where disturbances propagate; a comprehensive CAS theory must account for this fact.

3 A DPE Simulation Model

To further investigate the DPE process and the role of disturbances and connectivity in CAS we created an abstract model of resource flow through a network. We briefly discuss the model and some preliminary results here. The main objective of this paper is to review the empirical evidence for DPE and to discuss its relationship to other CAS theories. The space limit does not permit us to examine the model in greater detail and more detailed results will be published elsewhere (paper in preparation).

The model consists of a number of nodes connected via directed edges. Energy flows along edges and nodes require energy to sustain themselves. All nodes in the system are designated "component nodes", except for one, designated the "source". The source node does not require energy, instead it produces a constant amount of energy at each iteration. Energy flows along downstream connections attached to a node. Each model iteration consists of three stages: energy propagation, node maintenance and structural modification.

Energy Propagation. At the start of each iteration each component node c passes a proportion of its stored energy f_c along its downstream connections. Total energy propagated downstream by c is $d_c = f_c \times (1 - r_c)$, where the retention factor $0 \leq r_c < 0$ is a random number drawn when c is created. The remaining energy $(f_c - d_c)$ is retained by the node. If c has no outgoing links, all of f_c is retained. Nodes at the end of downstream edges of c compete for the energy propagated by c. Competition for resources in real systems requires energy. This is modelled by a competition cost factor $k_c = 1 / (1 + e^{2 \times (l_c - i_c)})$, where l_c is the number of downstream edges from c, and $i_c > 0$ is a random number drawn when c is created, it is the maximum value of l_c such that most energy is not wasted by competition expenses. Each of the l_c downstream edges receives an equal amount of $(d_c \times k_c / l_c)$ units of energy from c. Any energy conversion in nature comes with a loss. To model this, every edge g has a flow efficiency value w_g associated with it, such that the amount of energy actually arriving at node c_q from node c_p is $u_{q,p} = (d_c \times k_c / l_c) - w_{g(p,q)}$, where $g(p, q)$ is the edge from c_p to c_q and $w_{g(p,q)}$ is a random number drawn when g is created.

Node Maintenance. After all nodes have propagated energy downstream, the total available energy f_c at each component node c is equal to the amount of energy retained by c during the propagation stage plus the sum of the incoming energy from all upstream edges. Every c has an associated maintenance cost $m_c > 0$ selected randomly when c is created. To maintain its existence, every c expends m_c energy units per iteration. If $m_c > f_c$, then c dies and is removed from the system along with all connected up- and down-stream edges. The source node never dies. If c accumulates a large amount of energy, it reproduces. This happens by creating a duplicate copy h of c. The offspring h receives the same number of edges as c. Each of these edges may be connected either to the respective partner of c, or to any other random node with equal probability, thus modelling random mutation. The reproduction process consumes an amount of energy significantly larger than m_c and remaining energy is divided evenly between c and h.

Structural Modification. Every iteration, a new component or a new edge is introduced into the network with a small probability. When a new component c_n is introduced, for every existing node p, an edge $g(p, n)$ is added with a small probability. New edges connect two randomly selected existing nodes. Similarly, nodes and edges are removed from the network with a small probability at each iteration simulating external disturbances.

The presented model captures major features of resource flow dynamics in several real-world CAS. For instance, the energy flow through food webs in ecosystems follows patterns very similar to those described here. Resource flow between primary and intermediate producers, and end-consumers in economies follows a similar pattern. Thus, the results obtained form our abstract model allow conclusions about a variety of CAS.

3.1 Results

Model dynamics explored under a range of parameter values coincide with the behaviour expected under the DPE framework. A detailed discussion is beyond the scope of this paper, but we briefly overview some of the results here. Some indicators of network dynamics are the number C of component nodes, the total amount E of energy stored by all component nodes in the system, and the network edge density D. The maximum node age A is an indicator on internal stability of the system.

In the absence of external disturbances (probability of random node and edge removal is zero), C and E are lower on average compared to cases with disturbances. This initially surprising result can be explained by the DPE process. In the absence of disturbances unfavourable configurations can only be removed through node starvation. In the presence of disasters that propagate through the system by cutting off nodes and reducing connectivity, the remaining network sub-structures exhibit more efficient and robust connectivity patterns. Additionally, newly created nodes can better compete with established nodes that stored significant amounts of energy when all nodes can equally be affected by disturbances. This increases potential for innovation and for discovery of even more stable configurations.

Another consistently emerging pattern is that low values of D strongly correlate with high values of C and E: a small number of connections is enough to efficiently

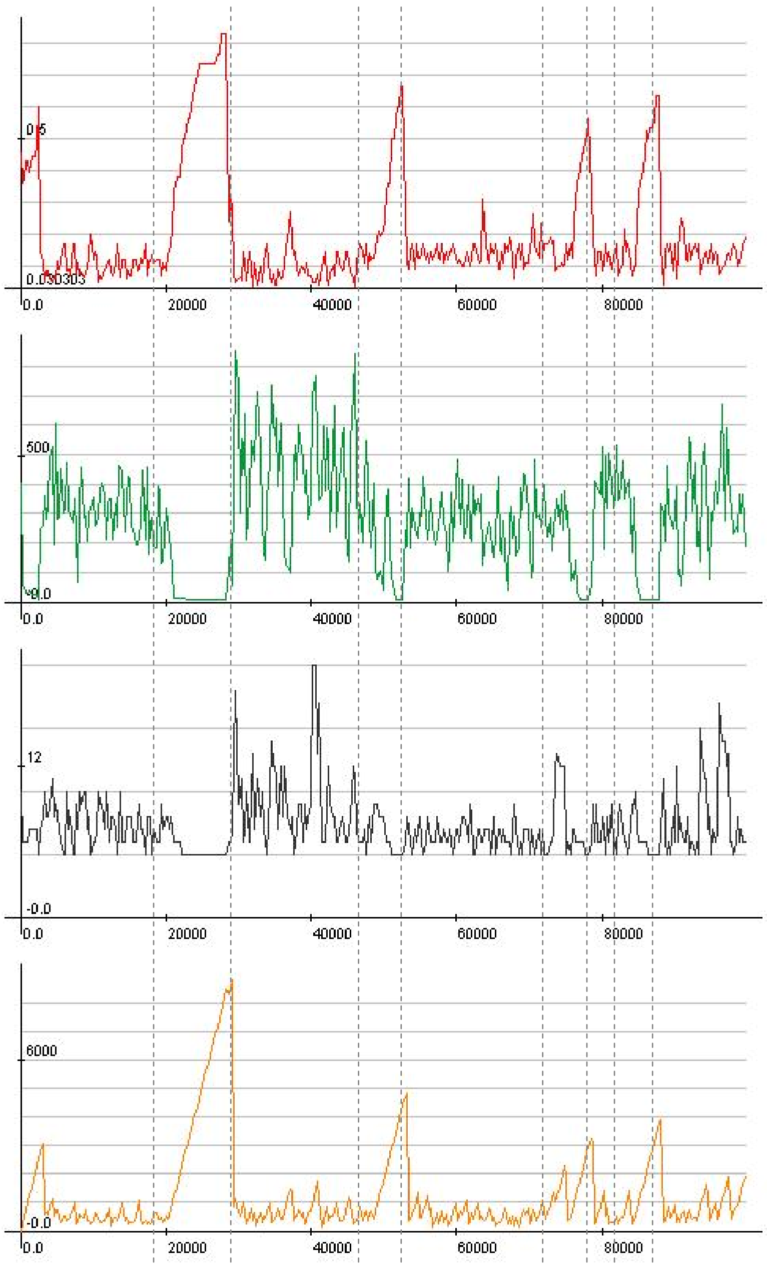


Fig. 3. A typical simulation run. Shown are (from top to bottom): edge density D, total stored energy E, number of component nodes C, oldest node age A. Mean node age (not shown) strongly correlates with A. The x-axes represent iterations. The vertical dashed lines are a visual aid to stress apparent phase changes.

distribute the energy across the components and additional edges lead to excessive energy expenditure due to unnecessary competition and flow friction along the edges (Fig. 3).

In a typical run A is normally low (< 1000), indicating internal instability. Over time, robust network configurations are discovered, signified by a growing value of A (>> 1000). Edge density in these stable configurations grows, making them less efficient and more susceptible to catastrophic change caused by structural modifications. Eventually, E reaches a very low value and the stable configurations collapse leading to the next variation phase (Fig. 3). This behaviour is in line with the predictions of DPE. However, in most of our experiments the variation phase was significantly longer than the DPE framework predicts. This observation may be explained by the absence of higher order stabilising control mechanisms such as selection between network configurations. Further experiments will test this conjecture.

4 Conclusions

Previous work has suggested that many interesting properties of CAS may be explained in terms of a network theoretical framework termed Dual Phase Evolution. According to DPE, networks underlying complex systems adapt and self-organise by alternately switching between two phases: a phase of high connectivity dominated by global component interactions and a phase of low connectivity dominated by local interactions.

We have presented evidence here that DPE may provide a causal explanation for known CAS properties that are typically expressed through other descriptive formalisms. The empirical data reviewed here imply that in many CAS, phase changes in network connectivity mediate dramatically different evolutionary conditions, contributing to their distinctive properties of self-organisation, perpetual novelty and evolution of modularity. Our simulation results indicate that DPE-like phase changes arise in a simple abstract model of resource flow in a network that is representative of a variety of systems. This work provides a step towards an integral understanding of CAS and suggests that more advances can be made by further empirical and theoretical studies of Dual Phase Evolution.

References

1. Bak, P.: How Nature Works: The Science of Self-Organized Criticality. Springer, Heidelberg (1999); Reprint edition
2. Bak, P., Tang, C., Weisenfeld, K.: Self-Organized Criticality. Physical Review A 38, 364–374 (1988)
3. Gunderson, L.H., Holling, C.S.: Panarchy: understanding transformations in human and natural systems. Island Press (2002)
4. Green, D.G.: Emergent Behaviour in Biological Systems. In: Green, D.G., Bossomaier, T.R.J. (eds.) Complex Systems: From Biology to Computation, pp. 24–33. IOS Press, Amsterdam (1993)

5. Green, D.G., Leishman, T.G., Sadedin, S.: Dual Phase Evolution: a mechanism for self-organization in complex systems. International Journal Complex Systems (2006)
6. Green, D.G., Newth, D., Kirley, M.G.: Connectivity and catastrophe - towards a general theory of evolution. In: Bedau, M., McCaskill, J.S., Packard, N.H., Rasmussen, S., McCaskill, J., Packard, N. (eds.) 7th International Conference on the Synthesis and Simulation of Living Systems, ALife VII (2000)
7. Green, D.: Fire and stability in the postglacial forests of southwest Nova Scotia. Journal of Biogeography, 29–40 (1982)
8. Willis, K., Bennett, K., Walker, D.: The evolutionary legacy of the Ice Ages–Introduction. Phil. Trans. R. Soc. Lond. B 359, 157–158 (2004)
9. Roshier, D., Robertson, A., Kingsford, R., Green, D.: Continental-scale interactions with temporary resources explain the paradox of large populations of desert waterbirds in Australia. Landscape Ecology 16, 547–556 (2001)
10. Swenson, N., Howard, D.: Clustering of contact zones, hybrid zones, and phylogeographic breaks in North America. The American Naturalist 166, 581–591 (2005)
11. Hewitt, G.: Genetic consequences of climatic oscillations in the Quaternary. Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 359, 183–195 (2004)
12. Avise, J., Walker, D.: Pleistocene phylogeographic effects on avian populations and the speciation process. Proceedings of the Royal Society B: Biological Sciences 265, 457–463 (1998)
13. Butlin, R., Walton, C., Monk, K., Bridle, J.: Biogeography of Sulawesi grasshoppers, genus Chitaura, using DNA sequence data. Biogeography and geological evolution of Southeast Asia, 355–359 (1998)
14. Eldredge, N., Gould, S.J.: Punctuated Equilibria: An Alternative to Phyletic Gradualism. Freeman Cooper, San Francisco (1972)
15. Gould, S.: The structure of evolutionary theory. Belknap Press (2002)
16. Gould, S., Eldredge, N.: Punctuated equilibrium comes of age. Shaking the Tree: Readings from Nature in the History of Life 17 (2000)
17. Paperin, G., Green, D., Sadedin, S., Leishman, T.G.: A Dual Phase Evolution Model of Adaptive Radiation in Landscapes. In: Randall, M., Abbass, H.A., Wiles, J. (eds.) ACAL 2007. LNCS (LNAI), vol. 4828, pp. 131–143. Springer, Heidelberg (2007)
18. Perkins, S.: Patterns from nowhere: Natural forces bring order to untouched ground. Science News 163, 314–316 (2003)
19. Kessler, M.A., Werner, B.T.: Self-organization of sorted patterned ground. Science 299, 380–383 (2003)
20. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
21. Cerný, V.: Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications 45, 41–51 (1985)
22. Ramamoorthy, C.V., Shekhar, S.: Stochastic backpropagation: a learning algorithm for generalizationproblems. In: 13th Annual International Computer Software and Applications Conference 1989 (COMPSAC 1989), Orlando, FL, USA, pp. 664–671 (1989)
23. Wang, X.H., Li, J.J.: Hybrid particle swarm optimization with simulated annealing. In: 2004 International Conference on Machine Learning and Cybernetics, vol. 4, pp. 2402–2405 (2004)

24. Liua, B., Wanga, L., Jina, Y.-H., Tangb, F., Huanga, D.-X.: Improved particle swarm optimization combined with chaos. Chaos, Solitons & Fractals 25, 1261–1271 (2005)
25. Cordon, O., Moya, F., Zarco, C.: A new evolutionary algorithm combining simulated annealing and genetic programming for relevance feedback in fuzzy information retrieval systems. Soft Computing 6, 308–319 (2002)
26. Lin, S.W., Lee, Z.J., Chen, S.C., Tseng, T.Y.: Parameter determination of support vector machine and feature selection using simulated annealing approach. Applied Soft Computing Journal 8, 1505–1512 (2008)
27. Sun, F., Sun, M.: Transductive Support Vector Machines Using Simulated Annealing. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS (LNAI), vol. 3801, pp. 536–543. Springer, Heidelberg (2005)
28. Alba, E., Dorronsoro, B.: Cellular Genetic Algorithms, vol. 42. Springer, Heidelberg (2008)
29. Whitley, L.D.: Cellular Genetic Algorithms. In: 5th International Conference on Genetic Algorithms. Morgan Kaufmann, San Francisco (1993)
30. Kirley, M.G.: A Cellular Genetic Algorithm with Disturbances: Optimisation Using Dynamic Spatial Interactions. Journal of Heuristics 8, 321–242 (2002)
31. Kirley, M., Li, X., Green, D.G.: Investigation of a cellular genetic algorithm that mimics landscape ecology. In: McKay, B., Yao, X., Newton, C.S., Kim, J.-H., Furuhashi, T. (eds.) SEAL 1998. LNCS (LNAI), vol. 1585, pp. 90–97. Springer, Heidelberg (1999)
32. Erdös, P., Rényi, A.: On the Evolution of Random Graphs. Matematikai Kutató Intézetének Közleményei 5, 17–61 (1960)
33. Green, D.G.: Self-Organization in complex systems. In: Bossomaier, T.R.J., Green, D.G. (eds.) Complex Systems, pp. 7–41. Cambridge University Press, Cambridge (2000)
34. Lenton, T.M., Van Oijen, M.: Gaia as a Complex Adaptive System. Philosophical Transactions of the Royal Society: Biological Sciences 357, 683–695 (2002)
35. Watson, A.J., Lovelock, J.E.: Biological homeostasis of the global environment: the parable of Daisyworld. Tellus B 35, 284–289 (1983)
36. Weber, S.L.: On Homeostasis in Daisyworld. Climatic Change 48, 465–485 (2001)
37. Holland, J.H.: Hidden Order: How Adaptation Builds Complexity. Perseus Books (1995)
38. Gavrilets, S.: Fitness Landscapes and the Origin of Species. Princeton University Press, Princeton (2004)
39. Paperin, G., Green, D.G., Leishman, T.G.: Dual Phase Evolution and Self-organisation in Networks. In: Li, X., Kirley, M., Zhang, M., Green, D., Ciesielski, V., Abbass, H.A., Michalewicz, Z., Hendtlass, T., Deb, K., Tan, K.C., Branke, J., Shi, Y. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 575–584. Springer, Heidelberg (2008)
40. Langton, C.G.: Computation at the edge of chaos: Phase transitions and emergent computation. Physica D: Nonlinear Phenomena 42, 13–37 (1990)
41. Langton, C.G.: Life at the Edge of Chaos. In: Langton, C.G., Taylor, C., Farmer, J.D., Rasmussen, S. (eds.) 2nd International Conference on the Synthesis and Simulation of Living Systems (ALife II). Addison-Wesley, Reading (1991)
42. Bak, P., Sneppen, K.: Punctuated equilibrium and criticality in a simple model of evolution. Physical Review Letters 71, 4083–4086 (1993)
43. Newman, M.E.J.: A model of mass extinction. Journal of Theoretical Biology 189, 235–252 (1997)
44. de Carvalho, J.X., Prado, C.P.C.: Self-Organized Criticality in the Olami-Feder-Christensen Model. Physical Review Letters 84, 4006 (2000)

45. Sornette, D., Johansen, A., Dornic, I.: Mapping Self-Organized Criticality onto Criticality. Journal de Physique I 5, 325–335 (1995)
46. Kinouchi, O., Prado, C.P.C.: Robustness of scale invariance in models with self-organized criticality. Physical Review E 59, 4964–4969 (1999)
47. Albert, R., Barabási, A.L.: Topology of Evolving Networks: Local Events and Universality. Physical Review Letters 85, 5234–5237 (2000)
48. Gleason, H.A.: Further views on the succession-concept. Ecology 8, 299–326 (1927)

Ant Colonies to Assign Terminals to Concentrators

Eugénia Moreira Bernardino[1], Anabela Moreira Bernardino[1], Juan Manuel Sánchez-Pérez[2], Juan Antonio Gómez-Pulido[2], and Miguel Angel Vega-Rodríguez[2]

[1] Research Center for Informatics and Communications, Department of Computer Science School of Technology and Management, Polytechnic Institute of Leiria, 2411 Leiria, Portugal
{eugenia.bernardino,anabela.bernardino}@ipleiria.pt
[2] Department of Technologies of Computers and Communications, Polytechnic School University of Extremadura, 10071 Cáceres, Spain
{sanperez,jangomez,mavega}@unex.es

Abstract. The last few years have seen a significant growth in communication networks. This has resulted in a large variety of new optimisation problems, most of them in the field of combinatorial optimisation. We address here the Terminal Assignment problem. The main objective is to minimise the cost links to form a network by connecting a collection of terminals to a collection of concentrators. In this paper we consider artificial Ant Colonies to assign terminals to concentrators. The algorithms use an improvement method to locate the global minimum. An Ant Colony algorithm is a swam-based optimisation algorithm that mimics the natural behaviour of ants. We show that artificial Ant Colonies are able to achieve feasible solutions to Terminal Assignment instances, improving the results obtained by previous approaches.

Keywords: Communication Networks, Optimisation Algorithms, Bees Algorithm, Terminal Assignment Problem.

1 Introduction

Terminal assignment (TA) is an important issue in telecommunication networks optimisation. The target of the TA problem implies fixing the minimum cost links to form a network between a specified set of terminals and concentrators [1, 2, 3]. Our purpose is to connect terminals to concentrators under three constraints:

1. each terminal is assigned to one (and only one) concentrator;
2. the total number of terminals assigned to any concentrator does not overload that concentrator, i.e. is within the concentrators capacity and;
3. balanced distribution of terminals among concentrators.

Under these constraints, an assignment with the lowest possible cost is sought.

The TA problem is a NP-complete combinatorial optimisation problem [1]. It means that the time required to solve the problem increases very quickly as the size of the problem grows. The intractability of this problem is a motivation for the pursuits of a metaheuristic that produces approximate, rather than exact, solutions. In [4, 5, 6] it was proposed the use of an Ant Colony Optimisation (ACO) algorithm as a new metaheuristic in order to solve combinatorial optimisation problems.

An ACO algorithm is essentially a Swarm Intelligence (SI) algorithm which simulates the natural behaviour of ants, including mechanisms of cooperation and adaptation. This new metaheuristic has been shown to be both robust and versatile. The ACO algorithm has been successfully applied to a range of different combinatorial optimisation problems [7].

An ACO algorithm is initialised with a population of individuals (i.e., potential solutions). These individuals are then manipulated over many iteration steps by mimicking the social behaviour of ants, in an effort to find the optimal solution in the problem solution space. A potential solution "walks" through the search space by modifying itself according to its past experience and its relationship with other individuals in the population and the environment [6].

This paper is a continuation of an earlier paper, which considered the use of a Hybrid ACO (HACO) algorithm to solve the TA problem [8]. Currently, we consider two New HACO (NHACO) algorithms based on the algorithm proposed by Bernardino et al. [8], but with some important modifications in the improvement method. The HACO uses pheromone trail information to perform modifications on TA solutions, unlike more traditional ant systems that use pheromone trail information to construct complete solutions. The HACO also uses a diversification mechanism that periodically reinitialises all the pheromone trails.

Embedded in the first NHACO algorithm we use a Local Search (LS) algorithm (NHACO-LS), which is used to improve the quality of the solutions. This LS method is similar to the LS used in HACO but we made some important improvements to reduce the computational time.

In the second NHACO algorithm we use a Tabu Search (TS) algorithm proposed by Bernardino et al. [9] (HACO-TS). The TS algorithm is a mathematical optimisation method, which belongs to the class of LS techniques.

We compare the performance of HACO-LS and HACO-TS with seven algorithms: classical Genetic Algorithm (GA), Tabu Search (TS) algorithm, Hybrid Genetic Algorithm (HGA), Genetic Algorithm with Multiple Operators (GAMO), Hybrid Differential Evolution (HDE) algorithm, Local Search Genetic Algorithm (LSGA) and Hybrid Ant Colony (HACO) algorithm, used in literature.

This paper is structured as follows. In Section 2 we describe the TA problem; in Section 3 we describe the implemented NHACO algorithms; in Section 4 we present the studied examples; in Section 5 we discuss the computational results obtained and in Section 6 we report about the conclusions.

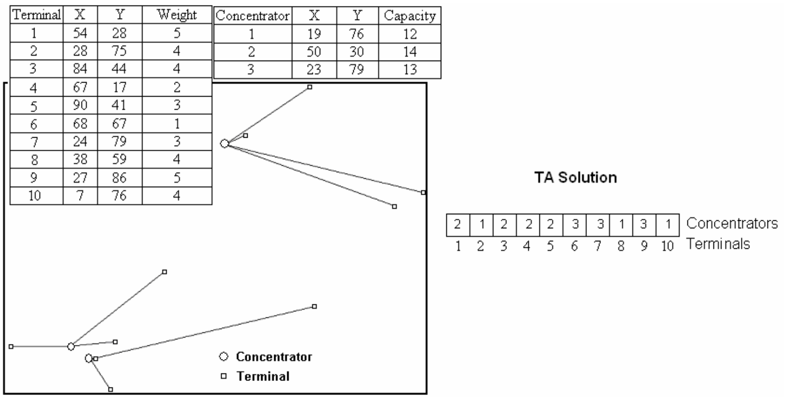
2 Terminal Assignment Problem

The TA problem involves the determination of which terminals will be serviced by each concentrator [1]. In the TA problem a communication network will connect N terminals with M concentrators. No terminal's demand exceeds the capacity of any concentrator. The TA Problem can be described as follows:

1. a set N of n distinct terminals;
2. a set M of m distinct concentrators;
3. a vector C, with the capacity required for each concentrator (each concentrator is limited in the amount of traffic that it can accommodate);
4. a vector T, with the capacity required for each terminal (the capacity requirement of each terminal is known and may vary from one terminal to another). The capacities are positive integers and T_i is smaller or equal to min (C_i...C_m);
5. a matrix CP, with the location (x,y) of each concentrator (the concentrators sites have fixed and known locations). The M concentrators are placed on the Euclidean grid;
6. a matrix CT, with the location (x,y) of each terminal (the terminals sites have fixed and known locations). The N terminals are placed on the Euclidean grid.

The first objective is to assign each terminal to one node of the set of concentrators, in a way that no concentrator oversteps its capacity. The second objective is to minimise the distances between concentrators and terminals assigned to them. Finally, the third objective is to ensure a balanced distribution of terminals among concentrators (see Section 3.3).

Fig. 1 illustrates an assignment to a problem with N=10 terminal sites and M=3 concentrator sites. The figure shows the coordinates for the concentrators, terminal sites and also their capacities.



Terminal	X	Y	Weight
1	54	28	5
2	28	75	4
3	84	44	4
4	67	17	2
5	90	41	3
6	68	67	1
7	24	79	3
8	38	59	4
9	27	86	5
10	7	76	4

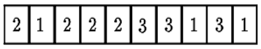
Concentrator	X	Y	Capacity
1	19	76	12
2	50	30	14
3	23	79	13

TA Solution

2	1	2	2	2	3	3	1	3	1	Concentrators
1	2	3	4	5	6	7	8	9	10	Terminals

Fig. 1. TA Problem – example

In this work, the solutions are represented using integer vectors. We use the terminal-based representation (see Fig. 2). Each position in the vector corresponds to a terminal. The value carried by the position *i* of the vector specifies the concentrator to which the terminal *i* is to be assigned.



2	1	2	2	2	3	3	1	3	1

Fig. 2. Terminal Based Representation

3 Ant Colonies

SI is an artificial intelligence technique involving the study of collective behaviour in decentralised systems [10]. Four widely known approaches are Ant Colonies, Particle Swarm Optimisation (PSO), Artificial Bee Colony (ABC) and Bees Algorithm. All these approaches can be used in real-world optimisation problems.

ACO algorithm is a SI algorithm that mimics the natural behaviour of ants [4]. ACO is a population-based optimisation method for solving hard combinatorial optimisation problems. ACO is based on the indirect communication of ants, mediated by pheromone trails. In a natural ant colony, ants indirectly communicate with each other by depositing pheromone trails on the ground and thereby influencing the decision processes of other ants. This simple form of communication between individual ants gives rise to complex behaviours and capabilities of the colony as a whole.

The first algorithm which can be classified within this framework was presented by Dorigo, Maniezzo and Colorni [5, 6], and Dorigo [4] and, since then, many diverse variants of the basic principle have been reported in literature.

The real ants behaviour is transposed into an algorithm by making an analogy between:

1. real ants search - set of feasible solutions to the problem;
2. amount of food in a source - fitness function;
3. pheromone trail - adaptive memory.

In real ant colony, while walking from food sources to the nest or from the nest to food sources, each ant deposits a pheromone on the ground. All ants can smell the pheromone while they walk. Therefore, more pheromone on the path will increase the probability of all ants to follow. In short, the best paths will receive a greater deposit of pheromones.

The pheromone trails in ACO serve as a distributed, numerical information which the ants use to probabilistically construct solutions to the problem being solved and which the ants adapt during the algorithm execution to reflect their search experience.

The essential trait of ACO algorithms is the combination of a priori information about the structure of a promising solution with a posterior information about the structure of previously obtained good solutions.

Any high performing metaheuristic algorithm has to achieve an appropriate balance between the exploitation of the search experience gathered so far and the exploration of unvisited or relatively unexplored search space regions. In ACO there are several ways of achieving such a balance, typically through the management of the pheromone trails. In fact, the pheromone trails induce a probability distribution over the search space and determine which parts of the search space are effectively sampled. The management of pheromone trails is the most important component of an ant system. Exploration is a stochastic process in which the choice of the component used to construct a solution to the problem is made in a probabilistic way. Exploitation chooses the component that maximises a blend of pheromone trail values and partial objective function evaluations.

The standard ACO algorithm uses pheromones trail information to construct complete solutions. Gambardella et al. [11] in their paper present a Hybrid Ant Colony System coupled with a local search (HAS_QAP), applied to the quadratic assignment problem (QAP). HAS-QAP uses pheromone trail information to perform modifications on QAP solutions. The HACO algorithm proposed by Bernardino et al. [8] also uses pheromone trail information to perform modifications on TA solutions, unlike traditional ant systems that use pheromone trail information to construct complete solutions.

In this paper we explore one of the most successful emerging ideas combining LS and TS with a population-based algorithm. HACO uses a modified ACO to explore several regions of the search space and simultaneously integrates a mechanism (improvement method) to intensify the search around some selected regions. NHACO-LS uses the same LS used by HACO but with some important improvements and NHACO-TS uses a TS algorithm to improve the solutions quality.

For the TA, the set of pheromone trails is maintained in a matrix P of size $N*M$, where the entry P_{ij} measures the desirability of assigning terminal i to concentrator j.

The simplest way to exploit the ants search experience is to make the pheromone update a function of the solution quality achieved by each particular ant. In HACO only the best solution found during the search contributes to pheromone trail updating [11]. This makes the search more aggressive and requires less time to reach good solutions. Moreover, this has been strengthened by an intensification mechanism. The intensification mechanics is used to explore neighbourhood more completely.

The algorithm also uses a diversification mechanism after a pre-defined number of S iterations without improving the best solution found so far. Gambardella et al. [11] have shown that pheromone trail reinitialisation, when combined with appropriate choices for the pheromone trail update can be very useful to refocus the search on a different search space region and avoid the early convergence of the algorithm.

The main steps of the HACO algorithm are given below:

```
Initialise Parameters
Initialise Solutions (ants)
Evaluate Solutions
Apply Improvement Method
Evaluate Solutions
Initialise Pheromone Trails
WHILE TerminationCriterion()
    FOR each Solution in Population
        Modify Solution using Pheromone Trails
        Apply Improvement Method
        Apply Intensification Mechanism
    Update Pheromone Trails
    Apply Diversification Mechanism
```

NHACO-LS and NHACO-TS use the same algorithm model.

The next subsections describe each step of the algorithm in detail.

3.1 Initialisation of Parameters

The following parameters must be defined by the user: (1) `NA` = number of ants; (2) `MI` = maximum number of iterations; (3) `Q` = value used to initialise the pheromone trails; (4) `q` = probability exploration/exploitation; (5) `x1` = pheromone evaporation rate; (6) `x2` = pheromone influence rate; (7) `NM` = number of modifications and (8) `S` = number of iterations (used to apply diversification mechanism).

3.2 Initialisation of Solutions

The initial solutions can be created randomly or in a deterministic form. The deterministic form is based in the Greedy algorithm proposed by Abuali et al. [1]. This algorithm assigns terminals to the closest feasible concentrator.

3.3 Evaluation of Solutions

The fitness function is responsible for performing this evaluation and it returns a positive number (fitness value) that reflects how good the solution is. The fitness function is based on: (1) the total number of terminals connected to each concentrator (the purpose is to guarantee a balanced distribution of terminals among concentrators); (2) the distance between the concentrators and the terminals assigned to them (the purpose is to minimise the distances between concentrators and terminals assigned to them); (3) the penalisation if a solution is not feasible (the purpose is to penalise the solutions when the total capacity of one or more concentrators is overloaded). The final purpose is to minimise the fitness function.

Fitness function:

$$fitness = 0{,}9 * \sum_{c=0}^{M-1} bal_c + \qquad (1)$$

$$0{,}1 * \sum_{t=0}^{N-1} dist_{t,c(t)} + \qquad (2)$$

$$Penalization \qquad (3)$$

$$bal_c = \begin{cases} 10 & if\left(total_c = round\left(\frac{N}{M}\right) + 1 \right) \\ 20 * abs\left(round\left(\frac{N}{M}\right) + 1 - total_c \right) & \end{cases} \qquad total_c = \sum_{t=0}^{N-1} \begin{cases} 1 & if\,(c(t) = c) \\ 0 & \end{cases}$$

$$Penalization = \begin{cases} 0 & if\,(Feasible) \\ 500 & \end{cases}$$

$$dist_{t,c(t)} = \sqrt{(CP[c(t)].x - CT[t].x)^2 + (CP[c(t)].y - CT[t].y)^2}$$

c(t)= concentrator of terminal t t = terminal c = concentrator
M = number of concentrators N = number of terminals

3.4 Improvement Method

An improvement method is applied to each solution in the initial set of solutions in order to reduce its cost, if possible. After solutions modification, the improvement method is also applied to improve the solutions quality.

In NHACO_LS we use the LS method proposed by Bernardino et al. [8]. The LS algorithm consists on applying a partial neighbourhood examination.

The most common and simplest way to generate a neighbour is to swap two terminals in the permutation. The size of the neighbourhood is `N*(N-1)/2` in this case, which would be large for large-scale problems. This would waste a lot of computing time. Other way to generate a neighbour is to assign one terminal to other concentrator. The size of the neighbourhood is `N*(M-1)`, which is also large for large-scale problems. In our implementation we generate a neighbour by swapping two terminals between two concentrators - `c1` and `c2` (randomly chosen). The algorithm searches for a better solution in the initial set of neighbours. If the best neighbour improves the actual solution, then the LS algorithm replaces the actual solution with the best neighbour. Otherwise, the algorithm creates another set of neighbours. In this case, one neighbour results on assigning one terminal of `c1` to `c2` or `c2` to `c1`. The neighbourhood size is `N(c1)*N(c2)` or `N(c1)*N(c2) + N(c1)+N(c2)`.

The LS algorithm consists on the following steps:

```
c1 = random (number of concentrators)
c2 = random (number of concentrators)
NN = neighbours of ACTUAL-SOL (one neighbour results of
     interchange one terminal of c1 or c2 with one terminal
     of c2 or c1)
SOLUTION = FindBest (NN)
IF fitness (ACTUAL-SOL) < fitness(SOLUTION)
    NN = neighbours of ACTUAL-SOL (one neighbour results of
         assign one terminal of c1 to c2 or c2 to c1)
    SOLUTION = FindBest (NN)
    IF fitness (SOLUTION < fitness(ACTUAL-SOL)
       ACTUAL-SOL = SOLUTION
ELSE
    ACTUAL-SOL = SOLUTION
```

The evaluation process is the most time-consuming step of the LS algorithm, which is usually the case in many real-life problems. Our LS procedure has some important improvements compared to the LS proposed by Bernardino et al. [8]. After creating a neighbour, the algorithm does not perform a full examination to calculate the new fitness value; it only updates the fitness value based on the modifications made to create the neighbour. The running time is considerably reduced.

In NHACO-TS a TS algorithm is applied to improve the solutions quality. The basic concept of TS was described by Glover [12]. The TS algorithm allows the search to explore solutions that decrease the objective function value only in those cases where these solutions are not forbidden. This is usually obtained by keeping track of the action used to transform one solution into the next. When an action is performed it is considered tabu for the next `K` iterations, where `K` is the tabu status length. A solution is forbidden if it is obtained by applying a tabu action to the current solution [13].

In our implementation, the TS only exploits a part of the neighbourhood. TS uses the same LS method described above to improve the solutions quality

The two concentrators (c1 and c2) which terminals are exchanged are classified as tabu attributes. A candidate can be chosen as a new current solution if the concentrators which terminals are exchanged are not the same as those in the tabu list. Normally in TS algorithm if a neighbour is the best solution found so far it could be selected as a move, even when it is tabu. In our implementation we don't explore neighbours when the two concentrators chosen are in the tabu list. In aspiration, just the best neighbour not tabu with a fitness value lower than the best is selected.

The TS ends when a maximum number of iterations is reached. Based on preliminary observations, we consider 5 iterations. With a higher value of iterations, the algorithm slows down. We also observed that a high number of iterations do not produce significant better results.

For the tabu list we consider N/20 elements. In the tests carried out with TS, it was verified that the number of elements in the tabu list does not have a significant influence on the efficiency and quality of the search. However, if the number of elements is high, the search space will be small, which may lead to a premature convergence of the algorithm. On the other hand, if the number of elements is small, the search space will be large, which may take a long time to obtain a good solution.

3.5 Pheromone Trails Initialisation

All pheromone trails P_{ij} are set to the same value $P_0=1/(Q*f(X^*))$ [11]. X^* is the best solution found so far and Q a parameter.

3.6 Modification of Solutions

It consists in repeating NM modifications. The modification is done assigning a terminal t to a concentrator c. First a terminal t is randomly chosen (between 1 and N) and then a concentrator c is chosen. A random number x is generated between 0 and 1. If x is smaller than q (parameter), the best concentrator c is chosen in a way that P_{tc} is maximum. This policy consists in exploiting the pheromone trail. If x is higher than q, the concentrator c is chosen with a probability proportional to the values contained in the pheromone trail. This is the mechanism to explore the solution space.

3.7 Intensification Mechanism

The intensification mechanism allows to explore the neighbourhood more completely and allows returning to previous best solutions. If the intensification is active and the solution X in the beginning of the iteration is better, the ant comes back to the initial solution X. The intensification is activated when the best solution found so far X^* has been improved and remains active while at least one ant succeeds on improving its solution during the iteration.

3.8 Pheromone Trails Update

To speed-up the convergence, the pheromone trails are updated by taking into account only the best solution found so far [11]. The pheromone trails are updating by setting:

$P_{ij}=(1-x1)*P_{ij}$, where $0<x1<1$ is a parameter that controls the evaporation of the pheromone trail.

$P_{ixi}* = P_{ixi}* + x2/fitness(X^*)$, where $0<x2<1$ is a parameter that controls the influence of the best solution X^* in the pheromone trail.

3.9 Diversification Mechanism

This mechanism restarts the pheromone trails and creates new solutions for each ant. For the following iteration, we kept the best solution found so far X^*.

3.10 Termination Criterion

The algorithm stops when a maximum number of iterations (MI) is reached.

More information about ACO can be found in ACO Website [7].

4 Studied Examples

In order to test the performance of our approach, we use a collection of TA instances of different sizes. We took 9 instances from literature [9].

Table 1 presents the 9 problems that we have used to test our algorithms. The first column represents the number of the problem (Problem) and the remaining columns show the number of terminals (N), the number of concentrators (M), the sum of the terminal's capacities (Total T), and the sum of the concentrator's capacities (Total C).

Table 1. TA Instances

Problem	N	M	Total T	Total C
1	10	3	35	39
2	20	6	55	81
3	30	10	89	124
4	40	13	147	169
5	50	16	161	207
6	50	16	173	208
7	70	21	220	271
8	100	30	329	517
9	100	30	362	518

5 Results

Bernardino et al. [8] studied the influence of the HACO parameters over the 9 instances of the TA problem. The best results obtained with HACO use NM between N/20 and N/3, $x1>0.4$ and $x2>0.4$, $Q=100$, S between N*2 and N*4, $q>0.4$ and NA={30,40} (see [8]). These parameters were experimentally found to be good and robust for the instances tested.

The number of ants NA has a significant influence on the execution time. Small populations are very desirable for reducing the required computational resources. The HACO algorithm has a good performance using initially a small population.

The number of modifications NM also has a significant influence on the execution time. In case of a high NM the resulting permutation tends to be too close to the best solution used to perform global pheromone trail updating, which makes it more difficult to generate new improving solutions. On the contrary, a small NM did not allow the system to escape from local minima, because after the improvement phase, the resulting solution was, in most cases, the same as the starting permutation.

To compare our results, we consider the results produced with the classical Genetic algorithm, Tabu Search algorithm, Hybrid Genetic algorithm, Genetic algorithm with multiple operators, Hybrid Differential Evolution algorithm, Local Search Genetic algorithm and Hybrid Ant Colony Optimisation algorithm.

The GA was first applied to TA by Abuali et al. [1]. In recent years, different GA algorithms have been applied to this problem [1, 2, 3, 9, 14, 17, 18]. The GA is widely used in literature to make comparisons with other algorithms. The adopted classical GA uses "one point" method for recombination, "change order" method for mutation and "tournament" method for selection. In "change order", two genes are randomly selected and exchanged.

In this paper, we only compare our algorithms with the algorithms proposed by Bernardino et al. [8, 9, 15, 17, 18], because they (1) used the same test instances; (2) adopted the same fitness function; (3) implemented the algorithms using the same language (C++) and; (4) adopted the same representation (terminal-based).

Table 2 and Table 3 present the best-obtained results with classical GA, TS, HGA, GAMO, HDE, LSGA, HACO, NHACO-LS and NHACO-TS. In both tables, the first column represents the number of the problem (P) and the remaining columns show the results obtained (BestF – Best fitness, Ts – Run Times) by the mentioned algorithms.

The algorithms have been executed using a processor Intel Core Duo T2300.

The initial solutions were created using the Greedy algorithm.

The Ts (Run Time) corresponds to the execution time that each algorithm needs to obtain the best feasible solution.

The values presented in tables 2 and 3 have been computed based on 100 different executions for each test instance.

Table 2. Results

P	GA		TS		HGA		GAMO		HDE		LSGA	
	BestF	Ts	BestF	Ts	BestF	Ts	BestF	Ts	BestF	Ts	BestF	Ts
1	65,63	<1s	65,63	<1s	65,63	<1s	65,63	<1s	65,63	<1s	65,63	<1s
2	134,65	<1s	134,65	<1s	134.65	<1s	134,65	<1s	134.65	<1s	134,65	<1s
3	284,07	<1s	270,26	<1s	270,26	<5s	270,26	1s	270,26	1s	270,26	<1s
4	286,89	<1s	286,89	<1s	286,89	<5s	286,89	1s	286,89	1s	286,89	<1s
5	335,09	<1s	335,09	<1s	335.09	<5s	335,09	1s	335,09	1s	335,09	<1s
6	371,48	1s	371,12	<1s	371,12	58s	371,12	1s	371,12	1s	371,12	1s
7	401,45	2s	401,49	1s	401,21	118s	401,21	2s	401,21	2s	401,21	1s
8	563,75	4s	563,34	1s	563,19	274s	563,19	8s	563,19	8s	563,19	7s
9	703,78	5s	642,86	2s	642,83	456s	642,83	8s	642,83	8s	642,83	7s

Table 3. Results – HACO, NHACO-LS and NHACO-TS

P	HACO		NHACO-LS		NHACO-TS	
	BestF	Ts	BestF	Ts	BestF	Ts
1	65,63	<1s	65,63	<1s	65,63	<1s
2	134.65	<1s	134.65	<1s	134.65	<1s
3	270,26	<1s	270,26	<1s	270,26	<1s
4	286,89	<1s	286,89	<1s	286,89	<1s
5	335,09	2s	335,09	<1s	335,09	<1s
6	371,12	3s	371,12	1s	371,12	<1s
7	401,21	4s	401,21	1s	401,21	1s
8	563,19	14s	563,19	7s	563,19	7s
9	642,83	25s	642,83	7s	642,83	7s

Tables 4 and 5 present the average fitnesses and standard deviations. The first column represents the number of the problem (Prob) and the remaining columns show the results obtained (AvgF – Average Fitness, Std – Standard Deviation) by the 10 algorithms.

To compute the results in tables 4 and 5 we use 300 iterations/generations for instances 1-4, 500 for the instance 5, 1000 for the instance 6, 1500 for the instance 7 and 2000 for instances 8-9.

The suggestions from literature helped us to guide our choice of parameter values for TS [9], HGA [9], GAMO [18], HDE [15], LSGA [17] and HACO [8].

For the TS, we consider a number of elements in the tabu list between 5 and 20. The parameters of HGA, GAMO and LSGA algorithms are set to crossover probability between 0.3 and 0.4, selection operator="tournament", mutation probability between 0.6 and 0.8, crossover operator="exchange terminals of two concentrators" and mutation operator= "multiple".

The parameters of the HDE algorithm are set to crossover probability between 0.3 and 0.4, factor F between 1.6 and 1.9 and strategy="Best1Exp".

The parameters of the HACO and NHACO algorithms are set to NA=30, S between 150 and 300, Q=100, q=0.9, x1=x2=0.8 and NM between 2 and 10.

The GA, HGA, GAMO, HDE and LSGA were applied to populations of 200 individuals.

The values presented in tables 4 and 5 have been computed based on 50 different executions (50 best executions out of 100 executions) for each test instance.

All algorithms reach feasible solutions for all test instances. The NHACO-LS and NHACO-TS algorithms can reach the best-known solutions for all instances. HDE, HGA, GAMO and HACO can also find the best–known solutions, but in a higher execution time. LSGA can reach the best-known solutions in a similar execution time. The TS algorithm is the fastest algorithm and can find good solutions in a reasonable running time. Comparing with HACO, NHACO algorithms are faster.

Since we are not trying to dynamically assign terminals to concentrators the running time isn't a significant parameter to determine the quality of the algorithms.

Table 4. Results – average fitnesses and standard deviations.

Prob	GA		TS		HGA		GAMO		HDE		LSGA	
	AvgF	Std	AvgF	Std	AvgF	Std	AvgF	Std	AvgF	Std	AvgF	Std
1	**65.63**	**0,00**	**65,63**	**0,00**	**65.63**	**0,00**	**65.63**	**0,00**	**65,63**	**0,00**	**65,63**	**0,00**
2	**134.65**	**0,00**	**134,65**	**0,00**	**134.65**	**0,00**	**134.65**	**0,00**	**134,65**	**0,00**	**134,65**	**0,00**
3	283,13	5,62	270,48	0,15	270,52	0,24	270,52	0,23	270,35	0,06	270,32	0,06
4	295,08	1,42	287,93	0,75	287,26	0,48	287,18	0,42	286,97	0,09	286,90	0,02
5	350,69	2,67	336,00	0,66	335,75	0,60	336,00	0,67	335,42	0,16	335,34	0,25
6	388,21	1,80	372,35	0,51	371,95	0,33	371,95	0,30	371,60	0,17	371,57	0,22
7	441,56	3,51	403,29	0,76	402,36	0,43	402,42	0,49	401,58	**0,12**	401,87	0,24
8	623,16	2,86	564,34	0,59	564,03	0,41	564,14	0,36	564,03	0,21	563,59	0,24
9	784,68	2,91	644,04	0,53	643,88	0,45	643,98	0,53	646,65	0,61	643,83	0,41

Table 5. Average fitnesses and standard deviations – HACO, NHACO-LS and NHACO-TS

P	HACO		NHACO-LS		NHACO-TS	
	AvgF	Std	AvgF	Std	AvgF	Std
1	**65,63**	**0,00**	**65,63**	**0,00**	**65.63**	**0,00**
2	**134,65**	**0,00**	**134,65**	**0,00**	**134.65**	**0,00**
3	270,32	0,06	270,32	0,06	**270,26**	**0,00**
4	286,91	0,04	286,91	0,04	**286,89**	**0,00**
5	335,11	0,03	335,11	0,03	**335,09**	**0,00**
6	371,55	0,17	371,55	0,17	**371,24**	**0,09**
7	401,61	0,15	401,61	0,15	**401,34**	**0,12**
8	563,55	0,16	563,55	0,16	**563,35**	**0,07**
9	643,67	0,38	643,67	0,38	**643,23**	**0,11**

The differences in terms of execution time are not significant. To establish which is the best algorithm we must observe the average quality of the produced solutions and the standard deviations.

As it can be seen in table 5, for larger instances the standard deviations and the average fitnesses for ACO algorithms are smaller. It means that the ACO algorithms are slightly more robust than GA, TS, HGA, GAMO, LSGA and HDE.

In NHACO-LS we only improve the source code to reduce the computational time. For that reason the average fitnesses and the standard deviations for HACO and NHACO-LS are exactly the same. The NHACO-TS presents a better average fitness and a smaller standard deviation for larger instances. All the statistics obtained show that the performance of NHACO-TS is superior in comparison with the algorithms studied.

6 Conclusions

In this paper we present artificial Ant Colonies to assign terminals to concentrators. The performance of the two proposed algorithms are compared with seven algorithms from literature, namely GA, TS, HGA, GAMO, LSGA, HDE and HACO. Ant

Colonies are swarm optimisation techniques, capable of performing simultaneous local and global search.

In comparison to other algorithms, artificial Ant Colonies achieve better results for the Terminal Assignment problem. The computational results show that artificial Ant Colonies had a stronger performance, improving the results obtained by previous approaches. Moreover, in terms of standard deviation, Ant Colonies also proved to be more stable and robust than the other algorithms.

Simulation results show that the proposed algorithms can produce satisfactory results regarding the solution quality and execution time for the Terminal Assignment problem.

For future work we propose the implementation of other Swarm Intelligence algorithms and the use of parallel algorithms to speed up the optimisation process.

References

1. Abuali, F., Schoenefeld, D., Wainwright, R.: Terminal assignment in a Communications Network Using Genetic Algorithms. In: Proc. of the 22nd Annual ACM Computer Science Conference, pp. 74–81. ACM Press, New York (1994)
2. Khuri, S., Chiu, T.: Heuristic Algorithms for the Terminal Assignment Problem. In: Proc. of the ACM Symposium on Applied Computing, pp. 247–251. ACM Press, New York (1997)
3. Salcedo-Sanz, S., Yao, X.: A hybrid Hopfield network-genetic algorithm approach for the terminal assignment problem. IEEE Transaction On Systems, Man and Cybernetics, 2343–2353 (2004)
4. Dorigo, M.: Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale (Optimisation, learning and natural algorithms). Doctoral dissertation, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy (1991)
5. Dorigo, M., Maniezzo, V., Colorni, A.: Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy (1991)
6. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics 26, 29–41 (1996)
7. Ant Colony Optimization HomePage, `http://iridia.ulb.ac.be/dorigo/ACO/ACO.html`
8. Bernardino, E., Bernardino, A., Sánchez-Pérez, J., Vega-Rodríguez, M., Gómez-Pulido, J.: A Hybrid Ant Colony Optimization Algorithm for Solving the Terminal Assignment Problem. In: International Conference on Evolutionary Computation (2009)
9. Bernardino, E., Bernardino, A., Sánchez-Pérez, J., Vega-Rodríguez, M., Gómez-Pulido, J.: Tabu Search vs Hybrid Genetic Algorithm to solve the terminal assignment problem. In: IADIS International Conference Applied Computing, pp. 404–409. IADIS Press (2008)
10. Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm intelligence. Morgan Kaufmann, San Francisco (2001)
11. Gambardella, L.M., Taillard, E.D., Dorigo, M.: Ant colonies for the quadratic assignment problem. Journal of the Operational Research Society 50(2), 167–176 (1999)
12. Glover, F.: Future paths for Integer Programming and Links to Artificial Intelligence. Computers and Operations Research 13(5), 533–549 (1986)

13. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Norwell (1997)
14. Yao, X., Wang, F., Padmanabhan, K., Salcedo-Sanz, S.: Hybrid evolutionary approaches to terminal assignment in communications networks. In: Recent Advances in Memetic Algorithms and related search technologies, vol. 166, pp. 129–159. Springer, Berlin (2005)
15. Bernardino, E., Bernardino, A., Sánchez-Pérez, J., Vega-Rodríguez, M., Gómez-Pulido, J.: A Hybrid Differential Evolution Algorithm for solving the Terminal assignment problem. In: International Symposium on Distributed Computing and Artificial Intelligence 2009, pp. 178–185. Springer, Heidelberg (2009)
16. Xu, Y., Salcedo-Sanz, S., Yao, X.: Non-standard cost terminal assignment problems using tabu search approach. In: IEEE Conference in Evolutionary Computation, vol. 2, pp. 2302–2306 (2004)
17. Bernardino, E., Bernardino, A., Sánchez-Pérez, J., Vega-Rodríguez, M., Gómez-Pulido, J.: Solving the Terminal Assignment Problem Using a Local Search Genetic Algorithm. In: International Symposium on Distributed Computing and Artificial Intelligence, pp. 225–234. Springer, Heidelberg (2008)
18. Bernardino, E., Bernardino, A., Sánchez-Pérez, J., Vega-Rodríguez, M., Gómez-Pulido, J.: A Genetic Algorithm with Multiple Operators for Solving the Terminal Assignment Problem. In: Nguyen, N.T., Katarzyniak, R. (eds.) New Challenges in Applied Intelligence Technologies, pp. 279–288. Springer, Heidelberg (2008)

A Statistical Study of the Effects of Neighborhood Topologies in Particle Swarm Optimization

Gregorio Toscano-Pulido, Angelina Jane Reyes-Medina, and José Gabriel Ramírez-Torres

CINVESTAV-Tamaulipas, Information Technology Laboratory. Scientific and Technological Park TECNOTAM – Km. 5.5 carretera Cd. Victoria-Soto La Marina Cd. Victoria Tamaulipas, 87130, Mexico
{gtoscano,areyes,grtorres}@tamps.cinvestav.mx
http://www.tamps.cinvestav.mx/

Abstract. The behavior of modern meta-heuristics is directed by both, the variation operators, and the values selected for the parameters of the approach. Particle swarm optimization (PSO) is a meta-heuristic which has been found to be very successful in a wide variety of optimization tasks. In PSO, a swarm of particles fly through hyper-dimensional search space being attracted by both, their personal best position and the best position found so far within a neighborhood.

In this paper, we perform a statistical study in order to analyze whether the neighborhood topology promotes a convergence acceleration in four PSO-based algorithms: the basic PSO, the Bare-bones PSO, an extension of BBPSO and the Bare-bones Differential Evolution. Our results indicate that the convergence rate of a PSO-based approach has a strongly dependence of the topology used. We also found that the topology most widely used is not necessarily the best topology for every PSO-based algorithm.

Keywords: Neighborhood topologies, Particle swarm optimization, Statistical test.

1 Introduction

Meta-heuristics have been proved to be very useful for solving optimization problems. The convergence velocity and the accuracy of these approaches for a given problem are usually directed by both: the variation operators and the values selected for the parameters of the algorithm (parameter setting). Therefore, researchers and other practitioners usually adopt values taken from the specialized literature which have been proved to work well on a wide range of problems. Nevertheless, these values may change depending on the problem at hand. Hence, the parameter setting plays a key role on the performance of any meta-heuristic. Tuning well these parameters is a hard problem, since they can usually take several values, and therefore, the number of possible combinations can be increased exponentially.

Kennedy & Eberhart [7] proposed an approach called "particle swarm optimization" (PSO) which was inspired on the choreography of a bird flock. The approach can be seen as a distributed behavioral algorithm that performs (in its more general version) multidimensional search. In the simulation, the behavior of each individual is affected by either, the best local or the best global individual.

Several PSO proposals have been developed in order to improve the performance of the original algorithm [4,7,8]. Such approaches have shown that the convergence behavior of PSO is strongly dependent on the values of the inertia weight, the cognitive coefficient and the social coefficient [1,12]. Other proposals have sought to eliminate the dependence of such parameters in order to avoid the parameter setting problem. Investigations within the particle swarm paradigm have found that the particles' interconnection topology interact directly with the function being optimized [3]. These studies have shown theoretically that the neighborhood topology affects (significantly) the performance of a particle swarm and that the effect depends on the function. Thus, some types of interconnection topologies can work well for some functions, while the same topologies can present problems with other test functions [3]. Despite the key role that the topology plays in PSO, it has been barely studied.

In this paper, we analyze whether the type of communication employed to interconnect the swarm accelerates or affects the algorithm convergence. In order to perform a wide study, we have selected six different neighborhood topologies: ring, fully connected, mesh, toroidal, tree and star; and a clustering algorithm: hierarchical. Such approaches were incorporated into four PSO versions: the basic PSO algorithm, the Bare-bones PSO (BBPSO) an extension of BBPSO called BBPSO(EXP) and the Bare-bones Differential Evolution (BBPSODE).

The remainder of the paper is organized as follows: Section 2 provides an overview of PSO and its variants used in this paper. Neighborhood topologies and clustering algorithms are presented in Section 3. Section 4 presents the description of our experiment and discusses the results obtained. Finally, Section 5 shows the concluding remarks and future work.

2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a stochastic, population-based optimization algorithm proposed by Kennedy and Eberhart in 1995 [5] which simulates the social behavior of bird flocks or school fish. In PSO, a swarm of particles fly through hyperdimensional search space being attracted by both, their personal best position and the best position found so far within a neighborhood. Each particle can be a solution to the optimization problem. The position of each particle is updated using equations (1) and (2), and the position of the best particle in its neighborhood is determined by the communication topology used [6,3].

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \tag{1}$$

$$\begin{aligned} v_{ij}(t+1) &= wv_{ij}(t) + c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t)) \\ &\quad + c_2 r_{2j}(t)(\widehat{y}_{ij}(t) - x_{ij}(t)), \end{aligned} \tag{2}$$

for $i = 1, \cdots, s$ and $j = 1, \cdots, n$

where w is the inertia weight [9], s is the total number of particles in the swarm, n is the dimension of the problem (i.e., the number of parameters of the function being optimized), c_1 and c_2 are the acceleration coefficients, r_{1j}, $r_{2j} \sim U(0,1)$, $\mathbf{x}_i(t)$ is the position of particle i at time step t, $\mathbf{v}_i(t)$ is the velocity of particle i at time step t, $\mathbf{y}_i(t)$ is the personal best position of particle i at time step t, and $\widehat{\mathbf{y}}_\mathbf{i}$ is the neighborhood best position of particle i at time step t.

Empirical and theoretical studies have shown that the convergence behavior of PSO is strongly dependent on the values of the inertia weight and the acceleration coefficients [13,1]. Wrong choices of such parameters may produce divergent or cyclic particle's trajectories. Several recommendations for values of such parameters have been suggested in the specialized literature [11], although these values are not universally applicable to every kind of problem.

A large number of PSO variations have been developed, mainly to improve the accuracy of solutions, diversity, convergence or to eliminate the parameters dependency [7]. Van den Bergh and Engelbrecht [13] and Clerc y Kennedy [1] proved that each particle converges to a weighted average of its personal best and neighborhood best position, that is,

$$\lim_{t \to +\infty} x_{ij}(t) = \frac{c_1 y_{ij} + c_2 \widehat{y}_{ij}}{c_1 + c_2} \tag{3}$$

This theoretically derived behavior provides support for the Bare-bones PSO (BBPSO). BBPSO was proposed by Kennedy in 2003 [4]. The BBPSO replaces the equation (1) and (2) by equation (4),

$$x_{ij}(t+1) = N\Big(\frac{y_{ij}(t) + \widehat{y}_j(t)}{2}, |y_{ij}(t) - \widehat{y}_j(t)|\Big) \tag{4}$$

Particle positions are therefore randomly selected from N which is a Gaussian distribution with: mean, equal to the average weighted of its personal best and the global best positions (i.e., the swarm attractor) and; deviation $y_{ij}(t) - \widehat{y}_j(t)$ which approximate zero as t increases.

Kennedy also proposed an alternative version of the BBPSO (EXP). He replaced equations (1) and (2) by equation (5),

$$x_{ij}(t+1) = \begin{cases} N\big(\frac{y_{ij}(t)+\widehat{y}_j(t)}{2}, |y_{ij}(t) - \widehat{y}_j(t)|\big) & \text{if } U(0,1) \\ & > 0.5 \\ y_{ij}(t) & \text{otherwise} \end{cases} \tag{5}$$

Based on the above equation, there is a 50% chance that the j the dimension of the particle dimension changes to the corresponding personal best position. This version of PSO biases towards exploiting personal best positions.

Differential Evolution (DE) [10] has been successfully applied to solve complex optimization problems. Its velocity of convergence and accuracy are mainly due to its mutation step size and its selection procedure since the first does not favored any previously defined distribution and the latter behaves like a hill-climber algorithm. Both characteristics make DE an ideal meta-heuristic to be hibridized with. Omran et al., [8] combined BBPSO and DE algorithms. The resulting algorithm was called Barebones Differential Evolution (BBPSODE). Unlike other PSO approaches, BBPSODE is practically independent of the communication topology.

In BBPSODE, each particle updates its position using equation (6).

$$x_{ij}(t) = \begin{cases} p_{ij}(t) + r_{2j} \times (x_{i_1 j}(t) - x_{i_2 j}(t)) & \text{if } \mathcal{U}(0,1) > p_r \\ y_{i_3 j}(t) & \text{in other case} \end{cases} \tag{6}$$

where:

$$p_{ij}(t) = r_{1j}(t) y_{ij}(t) + (1 - r_{1j}(t)) \widehat{y}_{ij}(t) \tag{7}$$

con $i_1, i_2, i_3 \sim \mathcal{U}(1, \ldots, s)$, $i_1 \neq i_2 \neq i$, r_{1j}, $r_{2j} \sim \mathcal{U}(0,1)$ y p_r is the recombination probability.

3 PSO: Neighborhood Topologies

In PSO, each particle inside of the swarm belongs to a specific communication neighborhood. Therefore, it was natural that several studies were performed in order to determine whether the neighborhood topology could affect the convergence [3,6,2]. These studies relied on theoretical proposals and implementations of neighborhood topologies commonly used by PSO. In such studies, some neighborhood topologies have performed better than others [3]. However, only a few topologies and problems were tested at a time. Therefore, our hypothesis to perform this study was that the topology used in a particle swarm might affect the rate and degree to which the swarm is attracted towards a particular region.

Since this paper analyzes the ring, fully connected, mesh, toroidal, tree, star topologies shown in Figure 1 and the hierarchical clustering algorithm, we described them below:

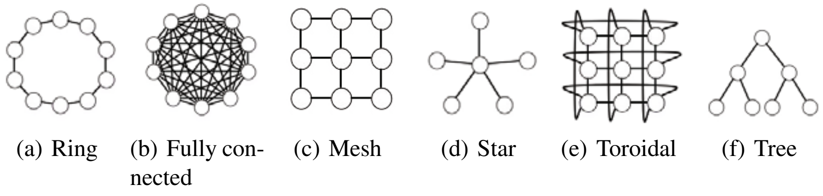

(a) Ring (b) Fully connected (c) Mesh (d) Star (e) Toroidal (f) Tree

Fig. 1. Neighborhood topologies used in this study

– Ring topology: This topology is also known as the *lbest* version in PSO (see Figure 1(a)). In this topology each particle is affected by the best performance of its k immediate neighbors in the topological population. In one common *lbest* case, $k = 2$, the individual is affected by only its immediately adjacent neighbors.

  In the ring topology, the neighbors are closely connected and thus, they react when one particle has a raise in its fitness, this reaction dilutes proportionally with respect to the distance. Thus, it is possible that one segment of the population might converge on a local optimum, while another segment of the population might converge to a different point or remain searching. However, the optima will eventually pull the swarm.

– Fully connected topology: Fully connected topology is also known as the full topology or the PSO gbest version (see Figure 1(b)). All nodes in this topology are directly connected among each other. In PSO this topology is also known as the PSO's *gbest* version, in which all particles in the entire swarm direct their flight toward the best particle found in the whole population (i.e., every particle is attracted to the best solution found by any member of the swarm). That is,

$$\widehat{\mathbf{y}}_i(t) \in \{\mathbf{y}_0(t), \mathbf{y}_1(t), \ldots, \mathbf{y}_s(t)\} = min\{f(\mathbf{y}_0(t)), f(\mathbf{y}_1(t)), \ldots, f(\mathbf{y}_s(t))\}, \quad (8)$$

Kennedy et al. suggested [7,6] that populations which use the *gbest* strategy tend to converge faster to an optima than those which use the *lbest* strategy, but also, they are more susceptible to converge to a local optima. However, the *gbest* topology is the most used by far.

– Star topology: In *star* topology, the information passes through only one individual (see Figure 1(d)). One central node influences and it is influenced by all other members of the population.

In this article, the central particle of the star topology is selected randomly. In each time step t, all particles of the entire swarm directs their flight toward one particle (the central particle), and the central particle directs its flight toward the best particle of the neighborhood. The star topology, effectively isolates individuals from each other, since information has to be communicated through the central node. This central node compares the performance of every individual in the population and adjusts its own trajectory toward the best of them. Thus the central individual serves as a kind of buffer or filter, slowing the speed of transmission of good solutions through the population. The buffering effect of the central particle should prevent premature convergence on local optima; this is a way to preserve diversity of potential problem solutions, though, it was expected that it might destroy the population's collaboration ability.

– Mesh topology: In this type of topology (see Figure 1(c)), one node is connected to several nodes, commonly each node is connected to four neighbors (in this case, we connect each node to the ones which are in the north, south, east and west of the particle's location).

In the mesh topology, the particles in the corners are connected with its two adjacent neighbors. The particles on the mesh's boundaries will have tree adjacent neighbors and the particles on the mesh's center will have four adjacent neighbors. Thus, there exists overlapping neighbors in each particle, allowing redundancy in the search process. The particles will be assigned to each node of the mesh from left to right and top-down. The mesh remains the same form in a single execution.

– Tree topology: It is also known as a hierarchical topology (see Figure 1(f)). This topology has a central root node (the top level of the hierarchy) which is connected to one or more individuals that are one level lower in the hierarchy (i.e., the second level), while each of the second level individuals that are connected to the top level central root individual will also have one or more individuals which are one level lower in the hierarchy (i.e., the third level) connected to it (the hierarchy of the tree is symmetrical).

The tree topology is constructed as a binary tree (using the particle's index as nodes), the root node is selected randomly among swarm and the remaining particles are distributed in the tree branches. The nodes (particles) in the tree must be, as possible, balanced in the tree branches. The root node searches for the best fitness obtained by their children (i.e., the second level) to redirected its flight. The second level nodes search for the best fitness found by both, children and parent, and so on.

– Toroidal topology: This topology is similar to the mesh topology, except that all particles in the swarm have four adjacent neighbors. As it is shown in Figure 1(e), the toroidal topology connects every corner particle with its symmetrical neighbor. The same occurs with the toroid boundaries. The assignment from particles to nodes will be similar to the mesh topology assignment.
– Clustering algorithms: Clustering is defined by the average number of neighbors that any two connected nodes have in common [3]. In PSO, the particles naturally cluster in more than one region of the search space usually indicate the presence of local optima. It seems reasonable to investigate whether information about the distribution of particles in the search space could be exploited to improve particle trajectories [7].

  We have implemented the hierarchical clustering algorithm:

  Hierarchical clustering considers the distance between one cluster and another cluster to be equal to the shortest distance from any member of one cluster to any member of the other cluster. If the data consists of similarities, then hierarchical clustering considers the similarity between one cluster and another cluster to be equal to the greatest similarity from any member of one cluster to any member of the other cluster.

4 Description of Our Experiment

We will compare the performance of the basic PSO, BBPSO, BBPSO (EXP), and BBPSODE algorithms discussed in Section 2. We have implemented: ring, full, star, mesh, toroidal and tree neighborhood topologies and the hierarchical clustering algorithm on each PSO algorithm. It is important to note that the present study focused on several swarm topologies, where connections were undirected, unweighted, and they do not vary over the course of a trial. The neighborhood topologies were constructed based on the index of each particle, then each particle has a unique identifier in the entire population. We also decided to study how a clustering algorithm can improve the performance of PSO. When the clustering algorithm were used, we use the euclidean distance as a measure and the connections were updated dynamically on each iteration of the trail.

For the basic PSO algorithm, we used $w = 0.72$ and $c_1 = c_2 = 1.49$. These values have been shown to provide good results [1,12,13].

For all the algorithms used in this section, the swarm size was $s = 50$. 200 iterations were perform by each algorithm (28 algorithms, since there were implemented 6 topologies + 1 clustering technique in 4 PSO variants). The resulting approaches were executed 30 independent runs. These values were used as defaults for all experiments which use static control parameters. Also, the distribution of the particles were 10×5 when the mesh and toroidal topologies were used. For the hierarchical clustering algorithm, 4 groups were asked for.

4.1 Test Functions

Nine test functions were selected from the specialized literature. Such test functions are described below:

A. *Sphere* function, defined as: $f(\mathbf{x}) = \sum_{i=1}^{N_d} x_i^2$,
where $\mathbf{x}^* = 0$ and $f(\mathbf{x}^*) = 0$ for $-100 \leq x_i \leq 100$
B. *Schwefel's* problem, defined as: $f(\mathbf{x}) = \sum_{i=1}^{N_d} |x_i| + \prod_{i=1}^{N_d} |x_i|$,
where $\mathbf{x}^* = 0$ and $f(\mathbf{x}^*) = 0$ for $-10 \leq x_i \leq 10$
C. *Step* function, defined as: $f(\mathbf{x}) = \sum_{i=1}^{N_d} (\lfloor x_i + 0.5 \rfloor)^2$,
where $\mathbf{x}^* = 0$ and $f(\mathbf{x}^*) = 0$ for $-100 \leq x_i \leq 100$
D. *Rosenbrock* function, defined as: $f(\mathbf{x}) = \sum_{i=1}^{N_d-1} (100(x_i - x_{i-1}^2)^2 + (x_{i-1} - 1)^2)$,
where $\mathbf{x}^* = (1, 1, \ldots, 1)$ and $f(\mathbf{x}^*) = 0$ for $-30 \leq x_i \leq 30$
E. *Rotated hyper-ellipsoid* function, defined as: $f(\mathbf{x}) = \sum_{i=1}^{N_d} (\sum_{j=1}^{i} x_j)^2$,
where $\mathbf{x}^* = 0$ and $f(\mathbf{x}^*) = 0$ for $-100 \leq x_i \leq 100$
F. *Generalized Schwefel* Problem 2.26, defined as: $f(\mathbf{x}) = -\sum_{i=1}^{N_d} (x_i \, sin(\sqrt{|x_i|}))$,
where $\mathbf{x}^* = (420.9687, \ldots, 420.9687)$ and $f(\mathbf{x}^*) = -4426.407721$ for $-500 \leq x_i \leq 500$
G. *Rastrigin* function, defined as: $f(\mathbf{x}) = -\sum_{i=1}^{N_d} (x_i^2 - 10cos(2\pi x_i) + 10)$,
where $\mathbf{x}^* = 0$ and $f(\mathbf{x}^*) = 0$ for $-5.12 \leq x_i \leq 5.12$
H. *Ackley's* function, defined as:
$f(\mathbf{x}) = -20exp\Big(-0.2\sqrt{\frac{1}{30} \sum_{i=1}^{N_d} x_i^2}\Big) - exp\Big(\frac{1}{30} \sum_{i=1}^{N_d} cos(2\pi x_i)\Big) + 20 + e$,
where $\mathbf{x}^* = 0$ and $f(\mathbf{x}^*) = 0$ for $-32 \leq x_i \leq 32$
I. *Griewank* function, defined as: $f(\mathbf{x}) = \frac{1}{4000} \sum_{i=1}^{N_d} x_i^2 - \prod_{i=1}^{N_d} cos\Big(\frac{x_i}{\sqrt{i}}\Big) + 1$,
where $\mathbf{x}^* = 0$ and $f(\mathbf{x}^*) = 0$ for $-600 \leq x_i \leq 600$

4.2 Discussion of Results

Since there are 28 algorithms and 9 test functions, then, it would be difficult to show numerical results. In order to present such results in a friendly-comparison way, we decided to present them as box-plot graphics. In Figures 3 and 2 are shown such results.

From these graphics, it is easy to see that BBPSODE presented the best behavior among the four algorithms, since it behaved similarly with all the topologies tried and it presented in 7 out of 9 test functions good results. However, when optimizing the *rotated hyper-ellipsoid* (see Figure 3(e)) and the generalized *Schwefel* problem (see Figure 2(a)), its behavior was not as good as the basic PSO.

Unlike BBPSODE, BBPSO (EXP) presented its best behavior with the fully connected and star topologies. Therefore, we can say that this approach has a highly dependence of the interconnection topology.

BBPSO behaved well, only second behind BBPSODE. The fully connected and star (see Figures 3(b) and 3(c)) were its best topologies.

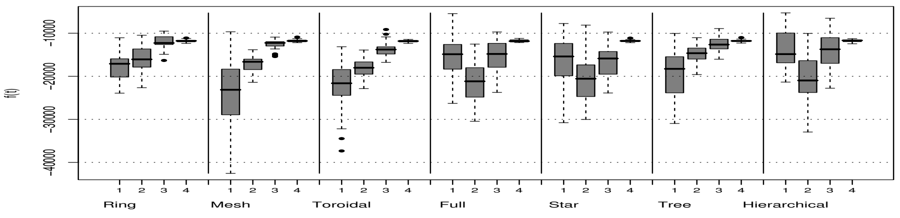
Although basic PSO was not the best approach, we can obtain important observations from this study:

1. Despite the fully connected (gbest) is the most popular topology, it produced the worst behavior in this study. Similar results were obtained when star topology was used.
2. Mesh and toroidal topologies were the topologies which produced the best results with basic PSO.

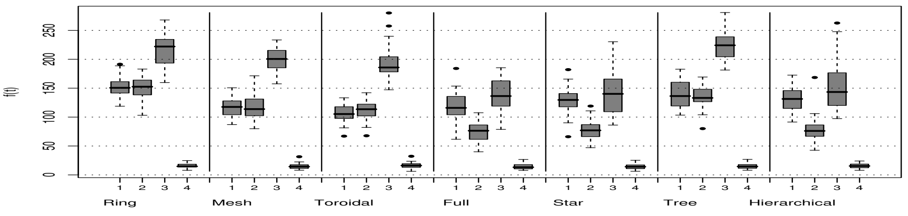
From our results, we can conclude that the topology plays a key role in PSO-based approaches.

4.3 Statistical Analysis

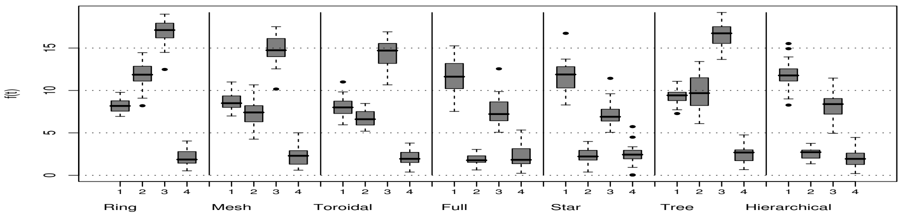
Although box-plot graphics allow us to visualize graphically the effect of the topologies in the performance of PSO algorithms and through them we can visually conclude which topology-algorithm performed better, it is important to conduct a statistical test that allows us to make more objective conclusions.



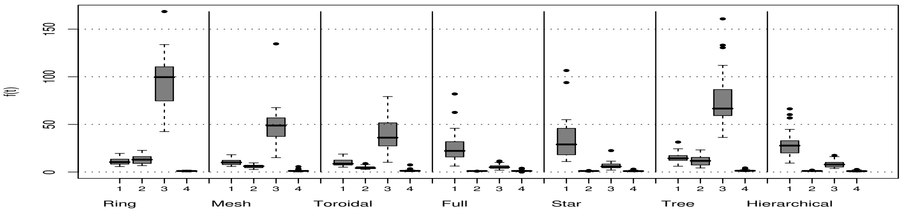
(a) Generalized Schwefel test function



(b) Rastrigin test function

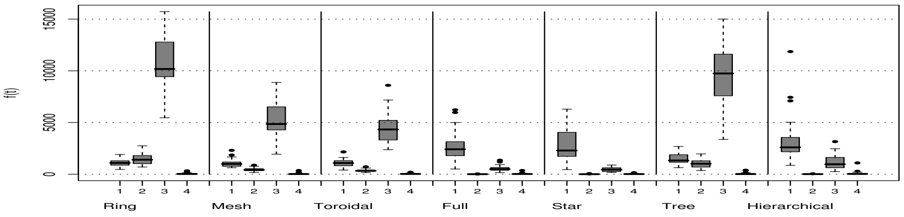


(c) Ackley test function

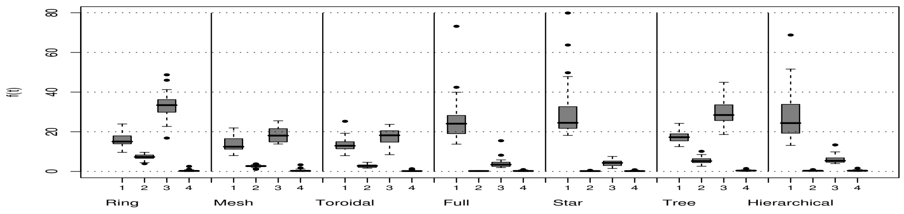


(d) Griewank test function

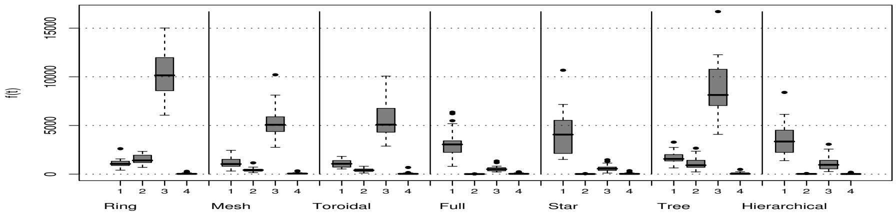
Fig. 2. Box-plots produced from the results of 30 independent runs: 1) PSO, 2) BBPSO, 3) BBPSO (EXP) and 4) BBPSODE



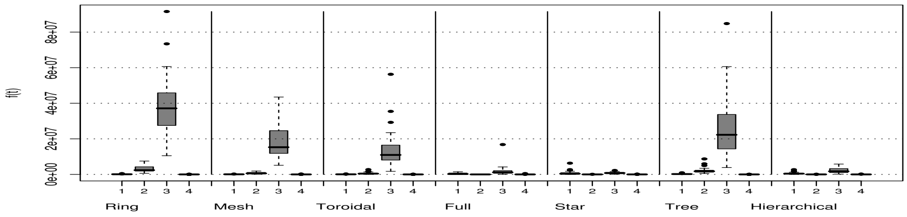
(a) Sphere test function



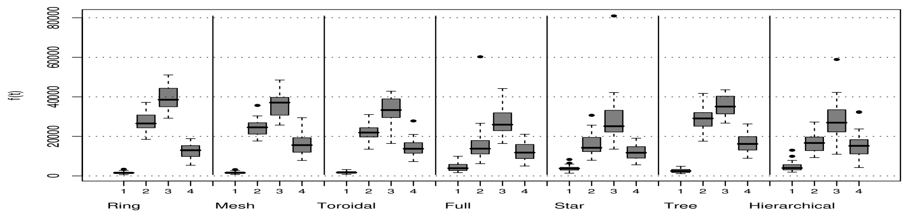
(b) Schwefel test function



(c) Step test function



(d) Rosenbrock



(e) Rotated hyper-ellipsoid function

Fig. 3. Box-plots produced from the results of 30 independent runs: 1) PSO, 2) BBPSO, 3) BBPSO (EXP) and 4) BBPSODE

In this way, it is possible to carry out a statistical analysis of the results through the t test in order to establish if: *the average difference between two topologies is significant or not*.

Since in this paper, we want to know if: *the interconnection topology affects the performance of a PSO algorithm*, then, from results previously described, it is clear that there is a difference in the performance of PSO when using topologies with different interconnection topology. But, we want to know whether this difference is statistically significant. Therefore, our null hypothesis is formulated as follows:

h_0: There is not any meaningful difference in the average response when using two different topologies.

h_1: There is a significant difference in the average response.

In order to apply the t test, we had to normalize the data, such that, we could compare two samples from different topologies. After that, we use the F statistical test in order to analyze the samples' variance. Finally, we applied the t test.

The results from this analysis are summarized in Tables 1 , 2, 3 and 4 for PSO, BBPSO, BBPSO (EXP) and BBPSODE, respectively.

In order to identify the topologies which produced the best results, follow the steps shown below:

– First, select the column which present the minimum average (since the functions in this paper are minimization problems) value in the headings of the table, right below the name name.
– Then, in such column, go down to a level below the main diagonal and cross the line to the right, all those topologies that had significance in their averages (those boxes that have the symbol $\surd$) will be the best performers topologies (since there is not any statistical difference among them).

4.4 Statistical Analysis: Discussion of Results

With the procedure described above, it is easy to identify the following conclusions:

– For the basic PSO approach (shown in the Table 1): the topology which perform better was the ring topology.
– For the BBPSO algorithm (shown in the Table 2): the topologies which showed better results were: star, fully connected and hierarchical clustering.
– For the BBPSO (EXP) algorithm (shown in the Table 3): the topologies that showed better results were: fully connected, star and the hierarchical clustering.
– For the BBPSODE algorithm (shown in the Table 4): it is difficult to determine the topology with best performance since all had significance in their averages, but it is possible to determine the topology which produced the worst results if we apply the above procedure with a slight modification: choose those topologies that did not present significance in averages (the boxes which have a $\times$ sign). On this basis, the worst behaved topologies were: mesh and tree.

Table 1. Statistical Analysis of Topologies in PSO Approach

Topology	Mesh 0.1702423	Toroidal 0.1597458	Full 0.332506	Star 0.3703755	Tree 0.2439002	Hierarchical 0.3715466
Ring 0.22063	$t_0 = 3.6542$ $p = 0.00027$ VD x	$t_0 = 4.5191$ $p = 6.95^{-06}$ VD x	$t_0 = -7.4339$ $p = 2.097e^{-13}$ VI x	$t_0 = -9.8472$ $p < 2.2e^{-16}$ VI x	$t_0 = -1.5859$ $p = 0.1131$ VD √	$t_0 = -9.8867$ $p < 2.2e^{-16}$ VI x
Mesh 0.170242		$t_0 = 0.9296$ $p = 0.3528$ VI √	$t_0 = -12.358$ $p < 2.2e^{-16}$ VI x	$t_0 = -15.0359$ $p < 2.2e^{-16}$ VI x	$t_0 = -5.8029$ $p = 8.488e^{-09}$ VI x	$t_0 = -15.0495$ $p < 2.2e^{-16}$ VI x
Toroidal 0.159745			$t_0 = -13.500$ $p < 2.2e^{-16}$ VI x	$t_0 = -16.2249$ $p < 2.2e^{-16}$ VI x	$t_0 = -6.815$ $p = 1.540e^{-11}$ VI x	$t_0 = -16.2308$ $p < 2.2e^{-16}$ VI x
Full 0.332506				$t_0 = -2.5921$ $p = 0.009663$ VI √	$t_0 = 6.3067$ $p = 4.101^{-10}$ VI x	$t_0 = -2.6614$ $p = 0.007894$ VI √
Star 0.370375					$t_0 = 8.8953$ $p < 2.2e^{-16}$ VI x	t_0 =-0.079 $p = 0.937$ VI √
Tree 0.2439						$t_0 = -8.9389$ $p < 2.2e^{-16}$ VI x

Table 2. Statistical Analysis of Topologies in BBPSO Approach

Topology	Mesh 0.3184229	Toroidal 0.2891353	Full 0.1127955	Star 0.1200146	Tree 0.503757	Hierarchical 0.1265078
Ring 0.5946722	$t_0 = 16.0759$ $p < 2.2e^{-16}$ VI x	$t_0 = 18.2095$ $p < 2.2e^{-16}$ VD x	$t_0 = 29.166$ $p < 2.2e^{-16}$ VD x	$t_0 = 28.6558$ $p < 2.2e^{-16}$ VD x	$t_0 = 5.0141$ $p = 7.176e^{-07}$ VI x	$t_0 = 28.1309$ $p < 2.2e^{-16}$ VD x
Mesh 0.3184229		$t_0 = 1.8167$ $p = 0.0698$ VI √	$t_0 = 12.9702$ $p < 2.2e^{-16}$ VI x	$t_0 = 12.4802$ $p < 2.2e^{-16}$ VI x	$t_0 = -10.5752$ $p < 2.2e^{-16}$ VI x	$t_0 = 12.0102$ $p < 2.2e^{-16}$ VI x
Toroidal 2891353			$t_0 = 11.4404$ $p < 2.2e^{-16}$ VI x	$t_0 = 10.9399$ $p < 2.2e^{-16}$ VI x	$t_0 = -12.5302$ $p < 2.2e^{-16}$ VI x	$t_0 = 10.4631$ $p < 2.2e^{-16}$ VI x
Full 0.11279955				$t_0 = -0.4756$ $p = 0.6346$ VI √	$t_0 = -23.1659$ $p < 2.2e^{-16}$ VI x	$t_0 = -0.8983$ $p = 0.3694$ VI √
Star 0.1200146					$t_0 = -22.6826$ $p < 2.2e^{-16}$ VI x	$t_0 = -0.4241$ $p = 0.6717$ VI √
Tree 0.503757						$t_0 = 22.1982$ $p < 2.2e^{-16}$ VD x

Table 3. Statistical Analysis of Topologies in BBPSO (EXP)

Topology	Mesh 0.4199163	Toroidal 0.3833513	Full 0.1480655	Star 0.1425615	Tree 0.5565467	Hierarchical 0.1861203
Ring 0.6176645	$t_0 = 12.11$ $p < 2.2e^{-16}$ VI x	$t_0 = 14.3236$ $p < 2.2e^{-16}$ VI x	$t_0 = 29.698$ $p < 2.2e^{-16}$ VI x	$t_0 = 30.0215$ $p < 2.2e^{-16}$ VI x	$t_0 = 3.5531$ $p = 0.0004131$ VI x	$t_0 = 25.6694$ $p < 2.2e^{-16}$ VI x
Mesh 0.4199163		$t_0 = 2.1991$ $p = 0.02828$ VI $\surd$	$t_0 = 16.8955$ $p < 2.2e^{-16}$ VI x	$t_0 = 17.224$ $p < 2.2e^{-16}$ VI x	$t_0 = -7.8267$ $p = 1.46e^{-08}$ VI x	$t_0 = 13.6939$ $p < 2.2e^{-16}$ VI x
Toroidal 0.3833513			$t_0 = 14.5962$ $p < 2.2e^{-16}$ VI x	$t_0 = 14.9259$ $p < 2.2e^{-16}$ VI x	$t_0 = -9.9059$ $p < 2.2e^{-16}$ VI x	$t_0 = 11.5334$ $p < 2.2e^{-16}$ VI x
Full 0.1480655				$t_0 = 0.3533$ $p = 0.724$ VI $\surd$	$t_0 = -24.0646$ $p < 2.2e^{-16}$ VI x	$t_0 = -2.2953$ $p = 0.02209$ VI $\surd$
Star 0.1425615					$t_0 = -24.3716$ $p < 2.2e^{-16}$ VI x	$t_0 = -2.6253$ $p = 0.008894$ VI $\surd$
Tree 0.5565467						$t_0 = 20.6841$ $p < 2.2e^{-16}$ VI x

Table 4. Statistical Analysis of Topologies in BBPSODE

Topology	Mesh 0.2352985	Toroidal 0.215506	Full 0.2069735	Star 0.1998743	Tree 0.2411042	Hierarchical 0.2226245
Ring 0.2098741	$t_0 = -1.4338$ $p = 0.1522$ VI $\surd$	$t_0 = -0.3306$ $p = 0.741$ VI $\surd$	$t_0 = 0.1682$ $p = 0.8665$ VI $\surd$	$t_0 = 0.6051$ $p = 0.5454$ VI $\surd$	$t_0 = -1.7791$ $p = 0.07576$ VI $\surd$	$t_0 = -0.7375$ $p = 0.4611$ VI $\surd$
Mesh 0.2352985		$t_0 = 1.0896$ $p = 0.2764$ VI $\surd$	$t_0 = 1.5425$ $p = 0.1235$ VI $\surd$	$t_0 = 2.0024$ $p = 0.04574$ VD x	$t_0 = -0.3112$ $p = 0.7557$ VI $\surd$	$t_0 = 0.5886$ $p = 0.4913$ VI $\surd$
Toroidal 0.2155060			$t_0 = 0.4824$ $p = 0.6297$ VI $\surd$	$t_0 = 0.92$ $p = 0.358$ VI $\surd$	$t_0 = -1.4228$ $p = 0.1553$ VI $\surd$	$t_0 = -0.4015$ $p = 0.6882$ VI $\surd$
Full 0.2069735				$t_0 = 0.4127$ $p = 0.68$ VI $\surd$	$t_0 = -1.8764$ $p = 0.06113$ VI $\surd$	$t_0 = 0. - 8727$ $p = 0.3832$ VI $\surd$
Star 0.1998743					$t_0 = -2.3544$ $p = 0.0189$ VI $\surd$	$t_0 = -1.3192$ $p = 0.1877$ VI $\surd$
Tree 0.2411042						$t_0 = 1.0136$ $p = 0.3112$ VI $\surd$

5 Conclusions and Future Work

In this paper, we implemented six different neighborhood topologies: ring, fully connected, mesh, toroidal, tree and star; and the hierarchical clustering algorithm in four PSO-based algorithms. Results indicate that the topology used affects the algorithm's performance. However, since some results were neither clear nor conclusive, then, we decided to use a statistical test.

Our main conclusions are the following:

- The use of toroidal topology promotes better convergence rates in the basic PSO algorithm.
- The use of the fully connected, star and hierarchical clustering approaches promote better convergence rates in the BBPSO and BBPSO (EXP) algorithms.
- Most of the topologies used performed similar for BBPSODE. Despite BBPSODE presented a similar performance when using the topologies tried when optimizing the nine test functions. However, it is clear that this approach did not perform well in generalized schwefel and rotated hyper-ellipsoid test functions.
- The topology most widely used (fully connected topology) did not produce good results in basic PSO whilst presented a good performance in BBPSO.
- Ring topology (which it is another topology widely used) presented a good convergence rate.
- The good selection of a topology can increase the performance of a PSO-based algorithm.

Some possible paths to extend this work are the following:

- Experiment with other PSO-based proposals and differential evolution algorithms.
- To include the parameter's values w, c_1 and c_2 in a similar study, in order to identify the relation among parameters (including the topology).

Acknowledgements. The first author gratefully acknowledges support from CONACyT through project 105060. The second author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Information Technology Laboratory at CINVESTAV-IPN. Also, this research was partially funded by project number 51623 from "Fondo Mixto Conacyt-Gobierno del Estado de Tamaulipas".

References

1. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
2. Jian, W., Xue, Y., Qian, J.: Improved particle swarm optimization algoritnms study based on the neighborhoods topologies. In: The 30th Annual Conference of the IEEE Industrial Electronics Society, IECON 2004, Busan, Korea, November 2004, vol. 3, pp. 2192–2196 (2004)

3. Kennedy, J.: Small worlds and mega-mind: Effects of neighborhood topology on particle swarm performance. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 1999, Washington, DC, USA, vol. 3, pp. 1931–1938 (1999)
4. Kennedy, J.: Bare bones particle swarms. In: Proceedings of the IEEE Swarm Intelligence Symposium, SIS 2003, April 2003, pp. 80–87. IEEE Press, Piscataway (2003)
5. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Joint Conference on Neural Networks, pp. 1942–1948. IEEE Press, Piscataway (1995)
6. Kennedy, J., Mendes, R.: Population structure and particle performance. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2002, pp. 1671–1676. IEEE Computer Society, Washington, DC, USA (2002)
7. Kennedy, J., Eberhart, R.C. (eds.): Swarm Intelligence. Morgan Kaufmann, San Francisco (2001)
8. Omran, M., Engelbrecht, A., Salman, A.: Bare bones differential evolution. European Journal of Operational Research 196(1), 128–139 (2008)
9. Shi, Eberhart, R.: A modified particle swarm optimizer. In: Proceedings of the IEEE Congress on Evolutionary Computation, Anchorage, AK, USA, May 1998, pp. 69–73 (1998)
10. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute (1995)
11. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Journal of Global Optimization 11(4), 359–431 (1997)
12. van den Bergh, F.: An Analysis of Particle Swarm Optimizers. PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa (November 2002)
13. van den Bergh, F., Engelbrecht, A.: A study of particle swarm optimization particle trajectories. Information sciences 176(8), 937–971 (2006)

Part III
Neural Computation

Genetic Algorithms Applied to Spectral Index Extraction

Diego Ordóñez[1,⋆], Carlos Dafonte[1], Minia Manteiga[2], and Bernardino Arcay[1]

[1] Information and Communications Technologies Department, Faculty of Computer Science
University of La Coruña, 15071, A Coruña, Spain
{dordonez,dafonte,cibarcay}@udc.es
[2] Department of Navigation and Earth Sciences, University of A Coruña,
15011 A Coruña, Spain
manteiga@udc.es

Abstract. Within the scope of computational astropysics, this work presents an experimental study on the application of genetic algorithms to the automated extraction of relevant information from stellar spectra. The input data are a dataset obtained through the collaboration of our research group with the Gaia project of the European Space Agency. The results show that predictions based on spectral indices, which in turn were extracted by means of genetic algorithms, have accuracy levels that are very similar to those obtained through wavelength information. Working with a reduced dataset also implies the reduction of complexity and increased performance.

Keywords: Genetic algorithm, Artificial neural network, Connectionist systems, FFT, Wavelet transform, GAIA mission, Stellar spectra, Stellar parameters.

1 Introduction

In the course of the last two decades, Observational Astrophysics has witnessed an authentic revolution in the capacities of telescopes and associated instruments as well as in the automation processes of data acquisition, processing, and archiving. Both the most recent earth telescope projects (e.g. the Great Canary Telescope at El observatorio del Roque de los Muchachos at La Palma, Spain), and the existing spatial telescopes (IUE, Hubble telescope, etc) include the creation of extensive databases, whose exploitation inevitably requires the use of automatic techniques for processing, classification, and parameterization.

The techniques that are used in Computational Astrophysics for this automatic processing of astronomic data (spectra, images, and fotometric data) are mainly twofold: statistical techniques (Minimal Distance Methods, Cluster Analysis), and techniques based on Artificial Intelligence Methods ([1], [2]), in particular Artificial Neural Networks (ANNs). A large number of publications are available in this field, such as those mentioned in this recent article by Allende ([3]), Bailer-Jones ([4], [5], [6]) or Gulati ([7]). In general terms, their purpose consists in identifying the astronomic source type

⋆ Spanish MEC project ESP2006-13855-CO2-02.

(star, galaxy, quasar, asteroid, etc), and, if the data set is uniform, as is the case with stars, characterize its members by parameterizing their main properties.

Our research group is a member of GAIA's scientific team, which was created to prepare the optimal algorithms that will allows us to carry out classification and parameterization tasks. The main objective is to determine the stellar atmospheric parameters, particularly effective temperatures, superficial gravities, metallicities, possible abundances of alpha elements, and individual abundances of certain chemical elements. The manipulation, analysis, and classification of all the information concerning the visible celestial bodies up to magnitudes $17 - 18$ is undoubtedly a challenge for both Astrophysicists and Computer and Artificial Intelligence Scientists.

In astronomy, great research efforts have been made in the development of automatic methods for prediction and classification. However, poor attention was paid to problem of selecting the most reliable features as inputs for those systems. This work tries to determine which is the most useful set of spectral features to be used as input of a learning algorithm. Our final goal is the prediction of the stellar atmospheric parameters (Teff, log g, [Fe/H] and [α/Fe]).

Genetic algorithms have been used in combination with artificial neural networks, with great success in many cases ([8], [9], [10]). In this study we combined both techniques in order to achieve an efficient solution to the problem of spectra parameterization. In contrast to the previously mentioned studies (the optimization of network parameters), the genetic algorithm in question was used to optimise input to the network (the selection of relevant characteristics of the input data). This work presents our first results on the automatic parameterization of atmospheric stellar parameters (RVS spectral region) using ANNs trained with synthetic stellar spectra and input optimized with genetic algorithms.

2 Signal Processing Techniques

The automatic techniques for classifying and parameterizing spectra are normally used in combination with some means of processing the signal prior to analysis. This process may have different goals: from reducing the dimensionality of the original signal (number of points), to a transformation required to explain certain features that were concealed in its original format.

The first transformation applied in this study is the discrete Wavelet transform. An efficient way of applying this transformation using filters was developed in 1988 by Mallat [11]. This filtering algorithm produces a fast Wavelet transform. We will refer to this process as a multilevel analysis, which in this case we will apply to spectra. In the wavelet analysis reference is made to approximations (low frequency components) and details (high frequency components). The concept of multilevel analysis refers to the repeated application of the filtering process to each of the successive approximations obtained in the signal, achieving a new level after each of these stages (Figure 1).

The experiment considers a total of three filtering levels, as shown in Figure 1. Three levels were chosen because as we descend to each level, the number of points for approximations and details is reduced by roughly half, and the approximations for lower levels signify very few points. In this Figure we may also see that by applying the

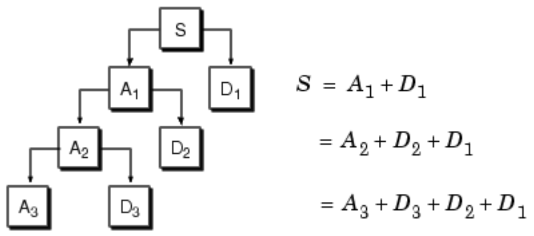


Fig. 1. Discrete Wavelet Transform. Multilevel decomposition

approximation and detail for a level, it is possible to obtain the approximation of the previous level by applying the inverse wavelet transform. This means that in order to carry out the experiment it is not necessary to consider all of the signals from the tree decomposition provided by the transform. Instead, we take the approximation from the lowest level and all of the details. By adding the number of points from all of these signals, we obtain a similar value to the number of points from the original signal.

Another of the pre-processing techniques frequently used for transforming stellar spectra is Principal Component Analysis, or PCA. The advantage of using this technique is that it reduces the dimensionality of the input data by eliminating variables with little information. It is used to determine the number of explanatory underlying factors of a series of data that explain its variability. Previous studies ([7]) have obtained worse results in analysing stellar spectra by applying PCA than by applying methods based on Wavelets ([12]).

PCA is an input-oriented analysis, meaning that it does not take into account the results we wish to obtain from the input (specific parameter). It also requires the involvement of an expert to decide how much typical deviation of the input data is represented in the selected variables once processing has been carried out. We aimed to predict four parameters based on a spectrum (temperature, gravity, metallicity and abundance of light elements), in the hope that the relevant points from the spectrum to ensure correct parameterization are different, depending on the case. For this reason we had to find a method that made it possible to reduce the dimensionality of the signal oriented to the parameter we wished to predict. The technique we chose in order to achieve these objectives was to use genetic algorithms. This technique will provide us with a selection of the relevant and specific points of the signal for each of the parameters we aim to obtain.

Also, considering the good results obtained based on the wavelet analysis applied to spectra [12], instead of applying the genetic algorithm technique directly on the signal, we have applied it to the result of applying the wavelet transform as previously described.

3 Data Description

For our tests the Gaia RVS Spectralib was used, a library of stellar spectra compiled by A. Recio-Blanco and P. de Laverny fron Niza Observatory, and B. Plez from Montpellier University. A technical note is available describing the models used for the

Table 1. Parameters and value ranges

Parameter	**Min**	**Max**
Teff	4500	7750
Logg	-0.5	5
[Fe/H]	-5	1
[α/Fe]	-0.2	0.4

atmospheres from which the synthetic spectra were calculated and which parameters were used ([13]). The library has a total of 9048 samples, the initial wavelength is 847.58 nm and the final 873.59 nm, the resolution is 0.0268 nm and the final number of points per signal is 971.

When the GAIA satellite becomes operative, the RVS instrument will inevitably include noise from various sources (sensitivity of the detectors, background noise near the source, instrumental noise, etc). We have therefore considered the possibility of working with synthetic spectra that are modified by various noise levels according to a simple model of noise, white noise, and various SNR values: 5, 10, 25, 50, 75, 100, 150, 200 and ∞.

The dataset represents the total number of examples that will be used to carry out the first stage of the experiment (comparison of results according to input domains). This set was arbitrarily divided into two subsets, in a proportion of 70%-30%; the first subset will be used to train the algorithms, the second for testing.

The above data were obtained through the participation of our research team in the GAIA project. The GAIA consortium has divided the tasks among several coordination units (CUs). Our research team belongs to CU8, the unit in charge of classification tasks, which means that we shall focus on classification through the parameterization of spectra from individual stars. Our input information consists of calibrated photometry, spectroscopy and astrometry, data gathered by the satellite and used to estimate the main astrophysical parameters of the stars: Teff, logg, [Fe/H], and [α/Fe].

4 Material and Methods

We aim to use the genetic algorithm as a selector for the characteristics of the signal (the spectrum) that contain relevant information in order to be able to predict a specific parameter. The genetic algorithm is coded using a chain of ones and zeros (binary alphabet) in which each gene (bit) represents one of the variables (points) from the input signal. In our case, this input signal will be the result of the wavelet transform described in section 2.

In order to represent the points of the signal that are selected by a specific individual, we use the genetic information of the chromosome as if it were a mask which, when applied to the input signal, will give us as a result the concatenation of the points for the inputs that are indicated in the mask with a 1. Those that contain a 0 will simply be rejected.

In order to carry out the tests with the genetic algorithms and neural networks, we used a rack containing 6 servers equipped with two Intel Xeon QuadCore processors and 16GB of RAM. For the automatic creation, training, evaluation and storage of the networks, we used the XOANE neural network tool ([14]), and in the case of the genetic algorithms we have developed software based on the Biojava library ([15]), open code software with a GNU licence. The Biojava library provides us with a framework for the implementation of the genetic algorithms, although the functions that comprise the behaviour of the algorithm were implemented by our research group. These functions are cross-over, mutation, selection and evaluation of individuals (fitness).

4.1 Genetic Algorithm Configuration

The configuration of the genetic algorithm comprises the specification of the strategies for selection, mutation, crosses and evaluation, as well as the specific parameters that govern their behaviour.

We applied a simple cross-over strategy in several points to be configured (in this study we tested configurations from one to three points), alternating the segments of information into which each of the parents is divided. The objective was to form two new individuals with the different segments that resulted from the selection of the cross-over points (justification explaining why we used this cross-over strategy).

Due to the high dimensionality of the individuals (having as many bits of information as the signal), if we consider all of the individuals in the population as candidates to be mutated, however low the probability of mutation, all of the individuals will be mutated at some stage. Also, if the probability is very low, only a few bits will be mutated, and the change will not be noticeable in the individual's fitness value. For this reason we reached a compromise by dealing with two probabilities for mutation: one that allows us to select the individuals from a population who will be mutated (mutation candidates), and another that allows us to determine if a gene is mutated or not at the moment of applying the operator. In this way, only a small number of individuals will be altered, and only a small (although potentially significant) portion of the information from the candidates to be mutated will be modified.

With regard to the selection function, we used the classic roulette algorithm, combined with an elitist strategy: determining the percentage of the best individuals that will form a part of the next generation. The usual selection operator is applied to the rest using the roulette method. We used this same strategy to determine the selection of the chromosomes for the population that will serve as a father, in order to combine their genetic information in the crosses.

The specific values of the parameters for applying the strategies described are shown in table 2. We carried out numerous trials with different parameter values. Those shown provide good results (see section 6), investing reasonable computation times. With regard to this aspect, we have two parameters that determine the total time invested in the execution of the genetic algorithm, which are the number of generations and the number of network training steps; this function represents practically all of the algorithm's workload.

The fitness function is a particular type of objective function that quantifies the goodness of a solution to a problem (chromosome) in a genetic algorithm, so that in this way

Table 2. Parameters and value ranges

Parameter	**Value**
Number of cross-overs	3
Mutation probability (one gene)	0.1
Mutation probability (individual)	0.3
Number of generations	100
Training steps	100
Elitist selection proportion	15%
Symbol probability	50,00%
Parental selection proportion	100,00%
Number of threads per node	8

each chromosome can be compared with the other components of the population. A fitness function is better the closer one comes to the intended objective. In our case, the objective was to discover the most relevant points from a spectrum in order to then train a neural network as optimally as possible. For this reason, the fitness function is based precisely on a network, and the fitness value is the mean of the total number of errors as an absolute value for the total number of selected tests (30%, see section 3).

Training a neural network to the point of achieving the optimum configuration of weights in which the network is considered to have been generalised is always a costly task, and as a result so is the process of computing the fitness function. In order to obtain results within a reasonable timescale, experience has shown us that after 100 training stages the network weights will provide us with a reliable orientation if the training maintains a constant trend towards the convergence minimum without any major fluctuations. For this reason the fitness value we have considered is the one obtained after completing this number of iterations. If we consider a larger number of iterations, we would expect to obtain a better result from the genetic algorithm, although we would have to accept the additional computing time involved.

Figure 2 shows the main stages of the genetic algorithm. We began by generating an initial population of 100 individuals or chromosomes, generating the population randomly using the mechanisms provided by the tool. Remember that we used a binary alphabet, with a symbol probability that was equal for all of the symbols, as shown in table 2. As a result, at first the number of points is reduced to half, leading to an accelerated training time and test time. We then carried out the initial evaluation of the individuals from the population, before iterating to obtain the successive populations. The next step formed a part of the iterative section: for the population resulting from the previous iteration, the chromosomes were selected that would form a part of the following population. As explained in this section, we applied the cross and mutation operators and evaluated the new individuals that were obtained, repeating the process until reaching the maximum number of generations.

4.2 Fitness Function and Artificial Neural Networks

Figure 3 shows a breakdown of the tasks carried out in the fitness function. This function began with the mask resulting from the genetic information of the individual we wished

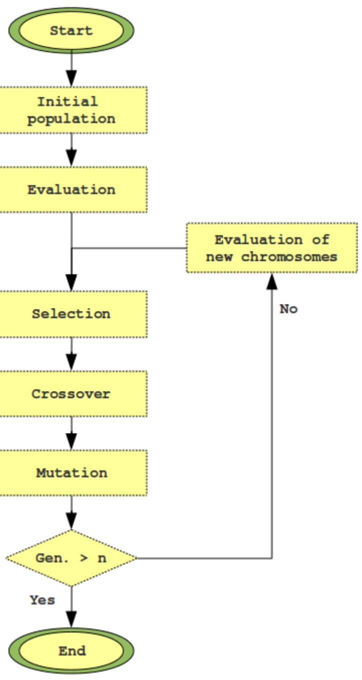


Fig. 2. Flow of the genetic algorithm

to evaluate, applying the transformed data (see section 4), and obtaining the training and test groups. The mask also provides us with information on the number of points that will comprise the input. Taking this number of points, and as the network will predict a single parameter, we then created a neural network. The architecture of the neural network is a feedforward with a single hidden layer, and the number of process elements from each layer depends on the inputs that the mask selects, as follows:

1. The number of process elements in the input that are equal to the number of points selected by the mask in the input signal.
2. For the hidden layer, we calculated the number of process elements as the minimum between 200 and the number of inputs divided by two. The number 200 was obtained based on experiments with the complete signal, with no more being required in order to obtain the generalisation point.
3. Number of outputs equal to a process element (a parameter to be predicted).

With regard to training, the online version of the error retropropagation algorithm was chosen. This training algorithm was chosen as a result of its proven use when applied to data of this kind derived from stellar spectra ([4], [16], [5]). In order to apply the algorithm a low learning rate was chosen (0.2), and 100 stages. The reason for the low learning rate is that we had a large number of patterns and the training process is carried out online, and so if we had used a high rate this would have led to excessive fluctuation of the weights.

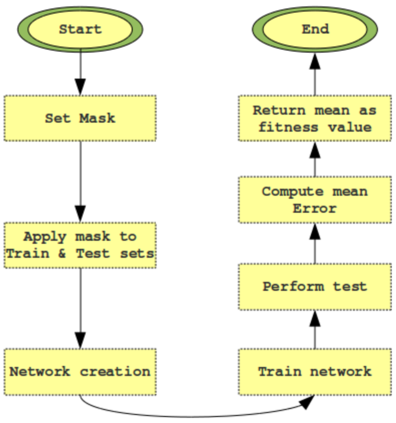


Fig. 3. Evaluation process of the fitness function

Once the network was created and as we already had the training and test groups, we then tested the network for the 100 stages described above (Section 4.1). Once training was completed we obtained the results for the tests as a whole, calculated the mean errors for the test as an absolute value, and established the inverse of this amount as the fitness value. It is important to take into account the fact that in this case, the fitness value is the inverse of the mean error so that the highest values signify better individuals.

5 Parallel Fitness Function

In the description of the input data (Section 3) we emphasised the large amount of data available and its dimensionality. The function that calculates the fitness of each of the individuals in the genetic algorithm is based on the training of the neural networks. Against this framework and considering the nature of the information being processed, the training of a network becomes a highly costly task in terms of time and computing resources. The sequential processing of the fitness functions on a computer is not viable in order to obtain results within a reasonable period of time. In this study we looked for a way of carrying out evaluations of the fitness functions for the new individuals in a parallel way, attempting to take full advantages of the computing power of the machines that were available (see section 4).

The parallel calculations in this case were based on the hardware features of the computers, each of which have two Quad Core processors. This characteristic makes it possible to launch concurrent threads that calculate the fitness function separately and independently from each other. Each of these threads is executed independently, although controlled centrally using a software module that acts as a pool. When the genetic algorithm decides to evaluate an individual, it sends the task to the pool, and if there are execution threads available it launches the fitness task. If at the moment of launching the fitness function the pool does not have any free threads, it queues the task until one is available. In this way, and in an ideal situation (not taking into account

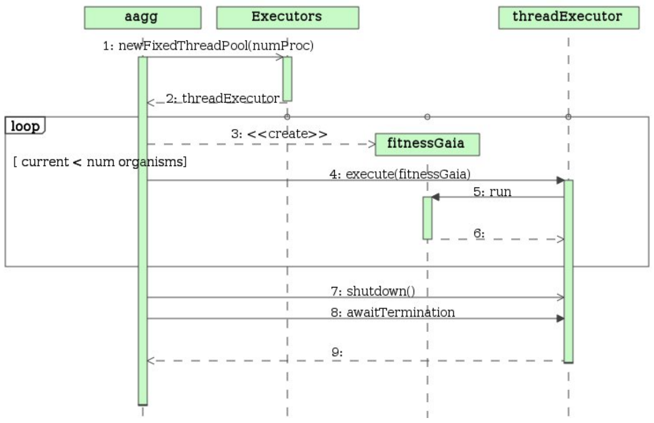


Fig. 4. Message sequence to invoke the evaluation of a chromosom

other bottlenecks in the application such as access to the shared memory bus), and considering the time dedicated to other times as minimal (crosses, mutations, selected etc.), we would divide the time required to pass from one generation to another by the number of threads available, and therefore also the total time spent on computing for the complete algorithm.

The way of interacting with the thread pool is as shown in Figure 4: fitnessGaia is the fitness function which in turn represents an execution thread. The genetic algorithm orders the execution of fitness through the threadExecutor object which plans the execution of the concurrent threads, queuing their execution if there are no free threads. In this Figure, after the loop zone, we can see that the genetic algorithm waits for the execution of all of the fitness functions to end before carrying out more tasks. It does so because the fitness value is necessary for the selection operator, which is the next operation to be carried out.

6 Results

After applying the genetic algorithm, the mask is obtained that will select the relevant information from the transformed signal, the result of applying the signal processing described in Section 2. The resulting series of data will provide us with the necessary data for the training and testing of a neural network, which this time we carry out in full (with 5000 training steps). As reference data for the training process we selected two sub-groups: clean spectra and SNR200, to which we applied the mask and network training. After training the networks we applied the mask to the reference group for the rest of the pattern collections, i.e. all of the noise levels considered, selecting the test patterns and calculating the results, as shown in tables 3 and 4. As may be seen, based on the results, executing the genetic algorithm with a certain degree of noise in the spectra makes it possible to obtain slightly better results in comparison to the same experiment carried out executing the genetic algorithm with clean spectra.

Table 3. Mean errors when selecting the points with the mask that results from applying genetic algorithms to clean spectra

	Teff	**logg**	**[Fe/H]**	**[α/Fe]**
SNR∞	91.3775	0.1614	0.110966	0.0640266
SNR200	116.185	0.209743	0.13699	0.0852009
SNR75	160.716	0.286768	0.202252	0.11027
SNR10	485.427	0.999177	0.590903	0.219483

Table 4. Mean errors when selecting the points with the mask that results from applying genetic algorithms to spectra with SNR200

	Teff	**logg**	**[Fe/H]**	**[α/Fe]**
SNR∞	73.8318	0.16255	0.113446	0.0604846
SNR200	101.58	0.211558	0.143434	0.0797694
SNR75	142.944	0.294903	0.200731	0.111127
SNR10	437.162	0.944661	0.590903	0.221791

Table 5. Typical deviation (σ) of the errors for all the parameters for the clean test spectra and SNR 75 case

	Teff	**logg**	**[Fe/H]**	**[α/Fe]**
SNR∞	93	0.172	0.120	0.0.074
SNR75	167	0.28	0.226	0.136

As the noise level increases, the results deteriorate. Despite this, they are especially relevant in the presence of noise, as we can compare them with the study [12] in which a comparison is made of different signal processing techniques applied to spectra. The advantage of this perspective is the reduction of the number of points in the signal and processing elements required to achieve a network that generalises and provides us with results with acceptable margins of error.

Another of the added advantages of this perspective for processing the information is that most of the errors are concentrated around zero, as may be seen in Figures 5 and 6, where there are also other examples with a higher error, but which represent less than 5% of the total. These Figures refer to the errors for the best case (clean spectra) and the effective temperature parameter. The concentration of the errors into small margins means that the algorithm is more robust, because in most of the cases we are sure of having good precision with a small margin of error. Table 5 shows additional information on this feature, showing the standard deviation of the errors. Each of the quantities shown should be studied within its context, as an error of one unit in temperature (degrees Kelvin) does not mean the same as an error in one unit in the case of gravity. These results may be considered with the study of Gulati and Ramírez [17] that analyses stellar spectra using genetic algorithms.

Figure 6 shows additional information, emphasising the fact that independently from the range of values of the parameter, the errors are highly concentrated around the correct value, and the fact that carrying out the analysis on a cold star (4000K) or a

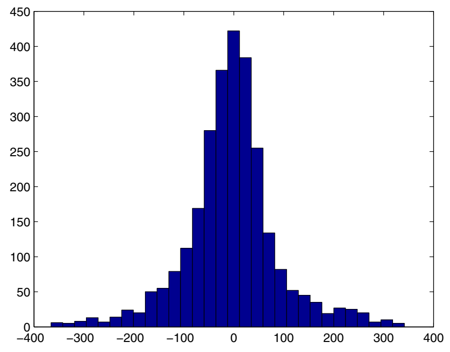


Fig. 5. Error dispersion for temperature

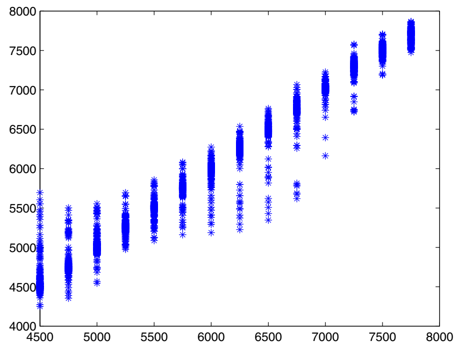


Fig. 6. Error dispersion for temperature per parameter value

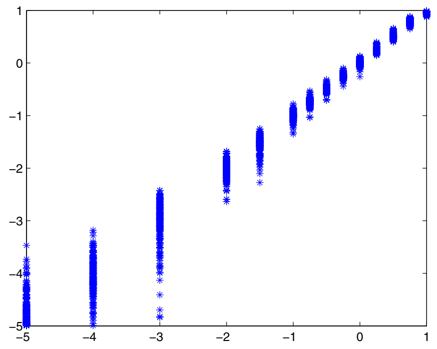


Fig. 7. Error dispersion for metallicity per parameter value

hot star (7750K) does not significantly influence the margins of error. This does not occur with all of the parameters; in the case of metallicity the opposite occurs for stars with low metallicity; the prediction is less reliable for stars with a high concentration of metallic elements, as may be seen in Figure 7. For the rest of the parameters the situation is similar to that of the effective temperature. As regards the dispersions with the presence of noise in the spectra, as would be expected the error is more distributed and flattened in the histogram shown in the Figure 5.

7 Conclusions

This work has demonstrated how genetic algorithms can be applied to the field of computational astrophysics. Genetic algorithms have proven to be a useful technique in a great many fields, including stellar spectra processing [17]. We used genetic algorithms and an artificial intelligence technique such as neural networks to process an input signal (the stellar spectrum) and select its relevant information for each parameter. Herein lies the fundamental difference with a statistical algorithm such as Principal Component Analysis, in which the relevant information is selected according to its variability without taking into account what it will be used for. It should also be noticed that we previously processed the signal based on discrete wavelet analysis, as described in Section 2.

The application of the genetic algorithm technique is mainly aimed at reducing the dimensionality of the signal, so that it may then reduce the time required to parameterise the spectra, obtaining the result from the neural network more quickly (due to the lesser complexity of the network in terms of processing elements). This aspect is of particular relevance in the GAIA mission, as already mentioned in section 1, as the aim is to classify millions of objects. Also, when carrying out training, the algorithm converges earlier as it only uses the information that is relevant in order to study the specific parameter it is dealing with.

Reviewing the results we found a robust approach to the parameterization of spectra, less demanding with regard to computing time. The combination of techniques allow us to use the advantages of both techniques: genetic algorithms (dimensionality reduction and information selection based on the parameter to predict) and neural networks (noise tolerance, good error rates and low error dispersion).

References

1. Rodríguez, A., Arcay, B., Manteiga, M., Carricajo, I.: An automated knowledge-based analysis and classification of stellar spectra using fuzzy reasoning. Expert Systems with Applications 27(2), 237–244 (2004)
2. Dafonte, C., Rodríguez, A., Arcay, B., Carricajo, I., Manteiga, M.: A comparative study of KBS, ANN and statistical clustering techniques for unattended stellar classification. In: Sanfeliu, A., Cortés, M.L. (eds.) CIARP 2005. LNCS, vol. 3773, pp. 566–577. Springer, Heidelberg (2005)
3. Von Hippel, T., Allende, C., Sneden, C.: Automated stellar spectral classification and parameterization for the masses. In: The Garrison Festschrift Conference Proceedings (June 2002)
4. Bailer-Jones, C.A.L.: A method for exploiting domain information in astrophysical parameter estimation. In: Astronomical Data Analysis Software and Systems XVII. ASP Conference Series, vol. 30 (2008)
5. Bailer-Jones, C.A.L.: Stellar parameters from very low resolution spectra and medium band filters. Astronomy and Astrophysics 357, 197–205 (2000)

6. Fiorentin, P.R., Bailer-Jones, C.A.L., Lee, Y.S., Beers, T.C., Sivarani, T., Wilhelm, R., Allende, C., Norris, J.E.: Estimation of stellar atmospheric parameters from sdss/segue spectra. Astronomy and Astrophysics 467, 1373–1387 (2007)
7. Harrinder, P., Gulati, R.K., Gupta, R.: Stellar spectral classification using principal component analysis and artificial neural networks. MNRAS 295, 312–318 (1998)
8. Hu, Y.-C.: Nonadditive grey single-layer perceptron with choquet integral for pattern classification problems using genetic algorithms. Neurocomputing 72, 331–340 (2008)
9. Rooij, A., Jain, L., Johnson, R.: Neural Network Training Using Genetic Algorithms. World Scientific Pub. Co Inc., Singapore (1996)
10. Kinnebrock, W.: Accelerating the standard backpropagation method using a genetic approach. Neurocomputing 91(3), 731–735 (1994)
11. Mallat, S.: A theory for multiresolution signal decomposition: The wavelet representation. Proc. IEEE Trans. on Pattern Anal. and Math. intel. 11(7), 674–693 (1989)
12. Ordóñez, D., Dafonte, C., Manteiga, M., Arcay, B.: Parameterization of rvs synthetic stellar spectra for the esa gaia mission: Study of the optimal domain for ann training. Expert Systems With Applications 37(2), 1719–1727 (2009)
13. Recio-Blanco, A., de Laverny, P., Plez, B.: Rvs-arb-001. European Space Agency technique note (2005)
14. Ordóñez, D., Dafonte, C., Arcay, B., Manteiga, M.: A canonical integrator environment for the development of connectionist systems. Dynamics of continuous, Discrete and Impulsive Systems 14, 580–585 (2007)
15. Down, T., Pocock, M.: The biojava project
16. Kaempf, T.A., Willemsen, P.G., Bailer-Jones, C.A.L., de Boer, K.S.: Parameterisation of rvs spectra with artificial neural networks first steps. In: 10th RVS Workshop, Cambridge (September 2005)
17. Gulati, R.K., Ramirez, F., Fuentes, O.: Prediction of stellar atmospheric parameters using instance-based machine learning and genetic algorithms. Experimental Astronomy 12(3), 163–178 (2001)

Algorithms of Image Restoration in Self-organizing Maps Grounded on Learning with Neighboring Inputs

Michiharu Maeda

Department of Computer Science and Engineering, Faculty of Information Engineering
Fukuoka Institute of Technology, 3-30-1 Wajiro-higashi, Higashi-ku, Fukuoka, Japan
maeda@fit.ac.jp

Abstract. Algorithms of image restoration in self-organizing maps are described grounded on learning with neighboring inputs. Novel approaches are presented that neighboring pixels as well as a notice pixel are prepared as an input and an original image is inferred according to an algorithm of self-organizing maps. The algorithm creates a map containing one unit for each pixel. Utilizing pixel values as input, image inference is conducted by self-organizing maps. An updating function with threshold based on the difference between input value and inferred value is introduced, so as not to respond to noisy input sensitively. The inference of an original image proceeds appropriately since any pixel is influenced by neighboring pixels corresponding to the neighboring setting. Experimental results are presented in order to show that our approaches are effective in quality for image restoration.

Keywords: Self-organizing maps, Image restoration, Degraded image, Neighboring inputs, Algorithm.

1 Introduction

Self-organizing neural networks realize the network utilizing the mechanism of the lateral inhibition among neurons with the local and topological ordering. The neighboring neurons would always respond for neighboring inputs [1,2]. For the localized inputs obviously, the outputs react locally. Huge amounts of information are locally represented and their expressions form a configuration with topological ordering. As an application of self-organizing neural networks, there are the combinatorial optimization problem, pattern recognition, vector quantization, and clustering [3]. These are useful when there exists redundancy among input data. If there is no redundancy, it is difficult to find specific patterns or features in the data. Although a number of self-organizing models exist, they differ with respect to the field of application. For self-organizing neural networks, the ordering and the convergence of weight vectors have been mainly argued [4]. The former is a topic on the formation of topology preserving map, and outputs are constructed in proportion to input characteristics [5,6]. For instance, there is the traveling salesman problem as an application of feature maps, which is possible to obtain fine results by adopting the elastic-ring method with many weights compared to inputs [7,8]. The latter is an issue on the approximation of pattern vectors, and the model expresses enormous information of inputs to a few weights. It is especially an important

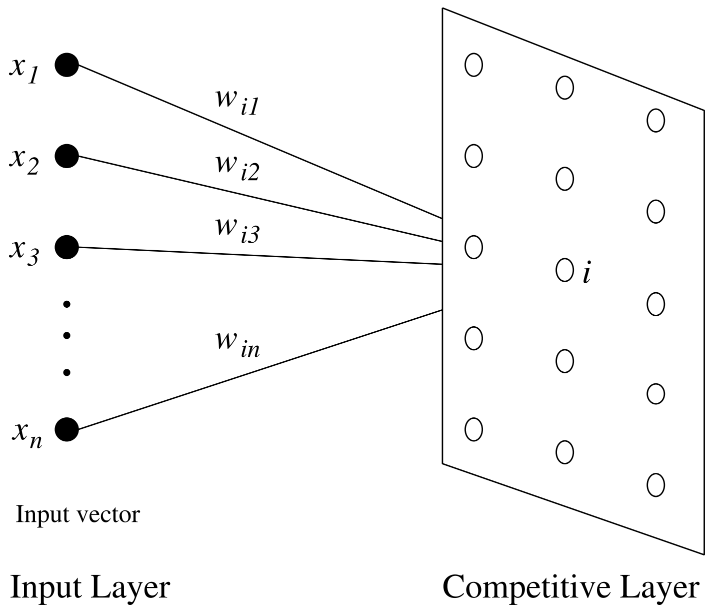


Fig. 1. Structure for self-organizing maps

problem for the convergence of weight vectors, and asymptotic distributions and quantitative properties for weight vectors have been mainly discussed when self-organizing neural networks are applied to vector quantization [9,10,11,12]. For image restoration, the smoothing methods, such as the moving average filter and the median filter, have been well known as a plain and useful approach [13]. From the standpoint of distinct ground, the inference of original image has been conducted by the model of Markov random field formulated statistically, based on the concept that any pixel is affected by neighboring pixels [14,15]. However their algorithm require a large number of iterations since they employ the stochastic model for statistical physics.

Algorithms of image restoration in self-organizing maps grounded on learning with neighboring inputs are described in this study. Novel approaches are presented that neighboring pixels as well as a notice pixel are prepared as an input and an original image is inferred according to an algorithm of self-organizing maps. Our model forms a map in which one element corresponds to each pixel. Image inference is conducted by self-organizing maps using pixel values as input. A renewal function with threshold is introduced in proportion to the difference between input value and inferred value. Based on this function, our approach does not respond to input including noise oversensitively. As any pixel is influenced by neighboring pixels corresponding to neighboring setting, the inference of an original image is appropriately promoted. Experimental results are presented in order to show that our approaches are effective in quality for image restoration.

2 Self-organizing Maps

Self-organizing maps realize the network with the local and topological ordering by utilizing the mechanism of the lateral inhibition among neurons. Neighboring neurons usually respond to the neighboring inputs. Huge amounts of information are locally represented and their expressions form a configuration with the topological ordering. For self-organizing maps, Kohonen's algorithm exists and is known as a popular and utility learning. In this algorithm, the updating of weights is modified to involve neighboring relations in the output array. The algorithm is applied to the structure as shown in Fig. 1. In the vector space R^n, an input $\boldsymbol{x}$, which is generated on the probability density function $p(\boldsymbol{x})$ is defined. The input $\boldsymbol{x}$ has the components x_1 to x_n. An output unit y_i is generally arranged in an array of one- or two-dimensional maps, and is completely connected to inputs via w_{ij}.

Let $\boldsymbol{x}(t)$ be an input vector at step t and let $\boldsymbol{w}_i(0)$ be weight vectors at initial values in R^n space. For given input vector $\boldsymbol{x}(t)$, we calculate the distance between $\boldsymbol{x}(t)$ and $\boldsymbol{w}_i(t)$, and select the weight vector as winner c minimizing the distance. The process is written as follows:

$$c = \arg\min_i \{\|\boldsymbol{x} - \boldsymbol{w}_i\|\}, \tag{1}$$

where arg$(\cdot)$ gives the index c of the winner.

With the use of the winner c, the weight vector $\boldsymbol{w}_i(t)$ is updated as follows:

$$\Delta\boldsymbol{w}_i = \begin{cases} \alpha(t)\,(\boldsymbol{x} - \boldsymbol{w}_i) & (i \in N_c(t)), \\ \boldsymbol{0} & \text{(otherwise)}, \end{cases} \tag{2}$$

where $\alpha(t)$ is the learning rate and is a decreasing function of time $(0 < \alpha(t) < 1)$. $N_c(t)$ has a set of indexes of topological neighborhoods for winner c at step t.

3 Image Restoration

When self-organizing maps are adapted to the traveling salesman problem, many weights are used in comparison with inputs. By disposing an array of one-dimensional map for output units, fine solutions on the basis of the position of weights after learning have been obtained approximately. In the meantime, when self-organizing maps are applied to vector quantization, a few weights are utilized in comparison with inputs for the purpose of representing huge amounts of information, and a number of discussions have been made on asymptotic distributions and quantitative properties for weight vectors.

In this section, a learning algorithm of self-organizing maps for image restoration is presented with the same number of both inputs and weights in order to infer an original image from a degraded image provided beforehand [16,17]. The purpose is to infer the original image from the degraded image containing random-valued impulse noise. Here, input χ as the degraded image and weight r_i as the inferred image are defined. A map forms that one element reacts for each pixel, and image inference is executed by self-organizing maps using pixel values as input.

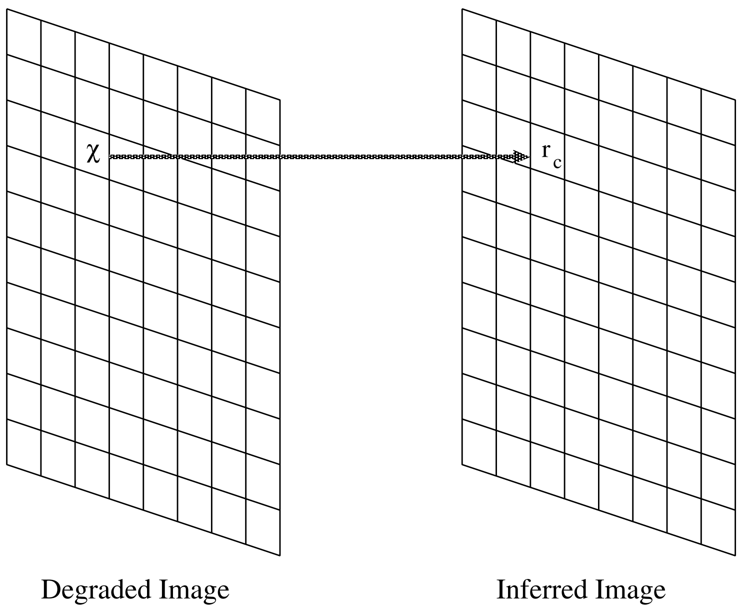


Fig. 2. Correspondence between degraded image and inferred image

To begin with, the value of r_i is randomly distributed near the central value of gray scale as initial value. Next, degraded image with $l \times m$ size is given. Input χ as pixel value is arbitrarily selected in the degraded image, and let r_c be a winner weight of the inferred image corresponding to χ. As shown in Fig. 2, both the positions χ and r_c agree under the degraded image and the inferred image. Therefore, inferred image r_i is updated as follows:

$$\Delta r_i = \begin{cases} \alpha(t)\Theta\left(\chi - r_i\right) & (i \in N_c(t)), \\ 0 & \text{(otherwise)}, \end{cases} \tag{3}$$

where $\Theta(a)$ is a function in which the value changes with threshold $\theta(t)$ presented as follows:

$$\Theta(a) = \begin{cases} a \ (|a| \leq \theta(t)), \\ 0 \ \text{(otherwise)}. \end{cases} \tag{4}$$

$\theta(t)$ is the difference between input χ and inferred image r_i in time t and the decreasing function of time, as $\theta(t) = \theta_0 - \lfloor \theta_0 t / T_{max} \rfloor$, where T_{max} is a maximum iteration and θ_0 is an initial threshold determined by trial and error as shown in numerical experiments. In the case of learning according to self-organizing maps, weights tend to react sensitively for noisy inputs. In order to avoid the tendency, Eq. (3) is adopted, instead of Eq. (2). By applying the function, neighboring pixels which obviously differ from the input value would be disregarded in learning. Using the functions, weights are updated until the termination condition is met. Image inference is appropriately promoted as given in the next section.

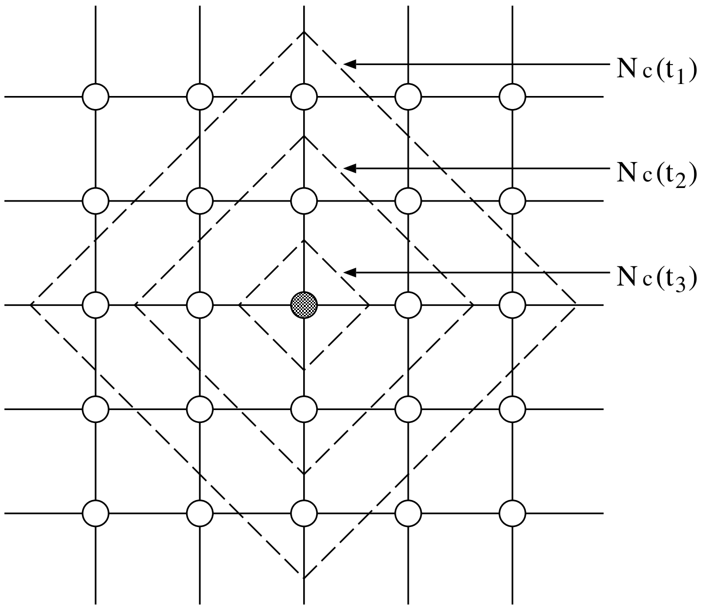


Fig. 3. Distribution of topological neighborhoods

Figure 3 shows an example of the arrangement of topological neighborhoods. The circle signifies the weight and the line which connects the circles denotes the topological neighborhood. In this figure, the black circle expresses the weight of winner c. As the set of topological neighborhoods changes $N_c(t_1)$, $N_c(t_2)$, and $N_c(t_3)$ when the time varies t_1, t_2, and t_3, respectively, it is shown that the number of topological neighborhoods decreases with time. By obtaining information of the neighboring pixels, it is possible to complement lost information about pixels based on the updating function.

Image restoration by self-organizing maps (IRS) algorithm is presented as follows.

[*IRS algorithm*]

```
Step 1 Initialization:
       Give initial weights $\{r_1(0), r_2(0), \cdots, r_{lm}(0)\}$ and maximum iteration $T_{max}$.
       $t \leftarrow 0$.
Step 2 Learning:
       (2.1) Choose input $\chi$ at random among $\{\chi_1, \chi_2, \cdots, \chi_{lm}\}$.
       (2.2) Select $r_c$ corresponding to input $\chi$.
       (2.3) Update $r_c$ and its neighborhoods according to Eq. (3).
       (2.4) $t \leftarrow t + 1$.
Step 3 Condition:
       If $t = T_{max}$, then terminate, otherwise go to Step 2.
```

Table 1. Variant models

Model	Input
I	Average of five pixels
II	Average of nine pixels
III	Median of five pixels
IV	Median of nine pixels

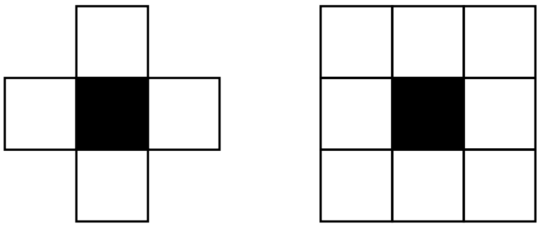


Fig. 4. Five pixels and nine pixels used in models I, II, III, and IV

In this study, a peak signal to noise ratio (PSNR) P is used as the quality measure after learning for image restoration. PSNR P is presented as follows [18]:

$$P = 10\log_{10}(\sigma/E) \ \ [\text{dB}] \tag{5}$$

where σ and E are the square of gray-scale length, i.e., $\sigma = (Q-1)^2$ as a gray scale Q and mean square error between the original image and the inferred image respectively.

In conventional approach, one pixel of the degraded image is given as an input. In this section, novel approaches are presented that a value of calculation for neighboring pixels as well as a notice pixel is prepared as an input, and the degraded image is restored according to self-organizing maps. For updating, we use the following equation.

$$\Delta r_i = \begin{cases} \alpha(t)\Theta\left(\gamma(\chi) - r_i\right) & (i \in N_c(t)), \\ 0 & (\text{otherwise}), \end{cases} \tag{6}$$

where $\gamma(\chi)$ is a function influenced by neighboring pixels.

With respect to $\gamma(\chi)$, four models are considered as the yielded input. Table 1 summarizes models I, II, III, and IV according to the standards of average of five pixels, average of nine pixels, median of five pixels, and median of nine pixels as the input, respectively.

In models I and III, five pixels are prepared as the input as shown in Fig. 4 (a). Models II and IV have nine pixels as the input (See Fig. 4 (b)). For models I and II, the input is an average of the notice pixel and the neighboring pixels. For models III and IV, the input is a median of the notice pixel and the neighboring pixels. According to four models, the inputs are changed for image restoration. By altering the inputs like these, the restored images which differ in quality for the image processing are constructed as shown in the next section.

Table 2. PSNR for results of MAF, MF, IRS, model I, model II, model III, and model IV. (Unit: dB).

	MAF	MF	IRS	Model I	Model II	Model III	Model IV
Image i	22.23	29.70	30.77	24.81	24.03	31.04	31.00
Image ii	21.65	28.36	28.08	23.92	23.29	29.42	29.40

4 Numerical Experiments

In the numerical experiments, image restoration is performed to infer the original image from a degraded image with the size 512×512 and gray scale 256. The degraded image contains 30% noise in comparison with the original image, namely, random-valued impulse noise, as shown in Fig. 5 (a). That is to say, the noise is included 30% pixels which are randomly chosen among the 512×512 pixels, and chosen pixels are given values from 0 to 255 at random. Initial weights are randomly distributed near the central value of gray scale Q. Parameters are chosen as follows: $l = 512$, $m = 512$, $Q = 256, T_{max} = 100 \cdot lm, N(t) = N_0 - \lfloor N_0 t/T_{max} \rfloor, \theta(t) = \theta_0 - \lfloor \theta_0 t/T_{max} \rfloor$, and $\alpha(t) = 0.01(1.0 - t/T_{max})$.

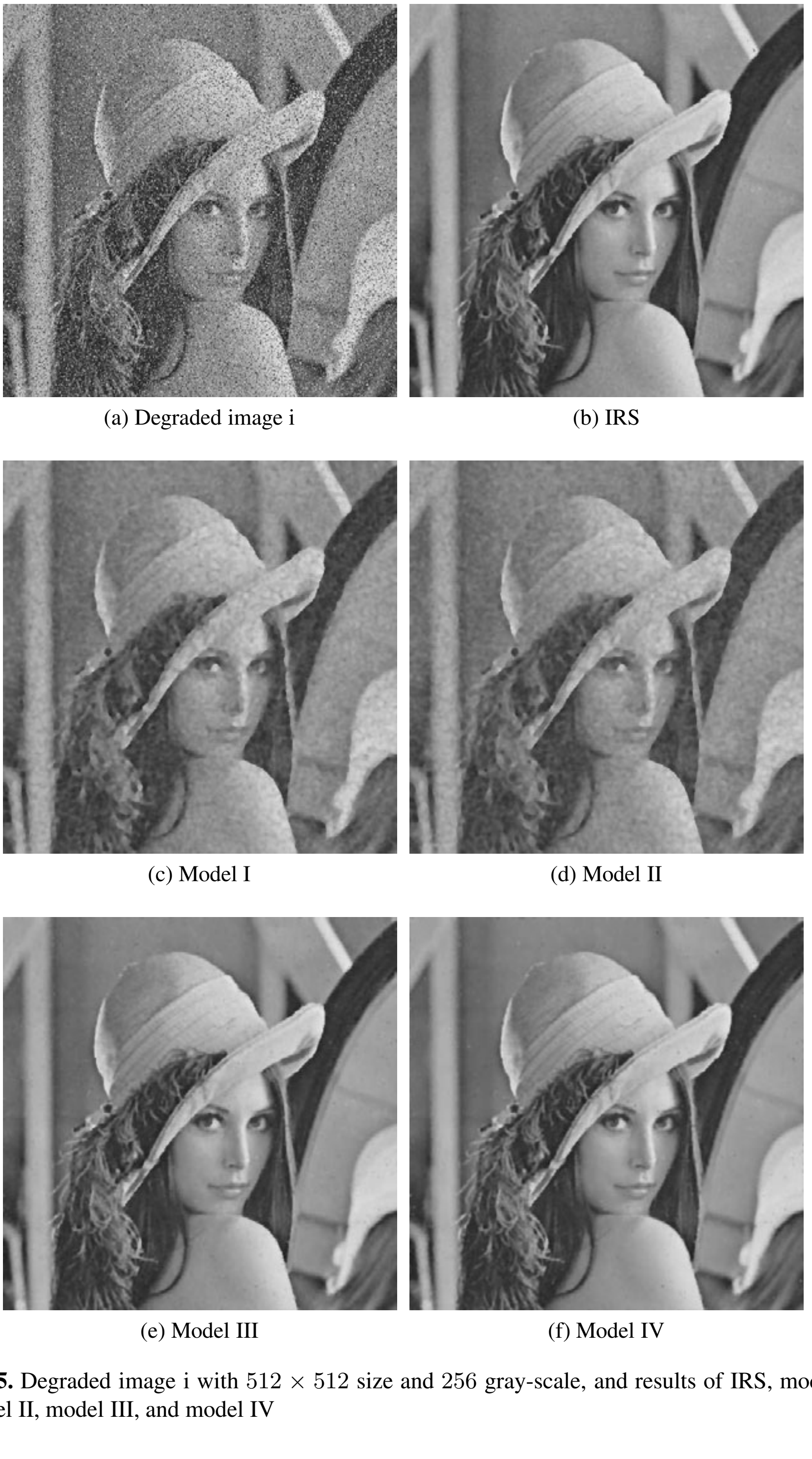
For image restoration, Fig. 5 (b), (c), (d), (e), and (f) show results of conventional model (IRS), model I, model II, model III, and model IV, respectively. The initial neighborhood and the initial threshold are $N_0 = 3$ and $\theta_0 = 95$ for IRS, $N_0 = 6$ and $\theta_0 = 69$ for model I, $N_0 = 7$ and $\theta_0 = 73$ for model II, $N_0 = 3$ and $\theta_0 = 96$ for model III, and $N_0 = 2$ and $\theta_0 = 98$ for model IV, respectively. According to the technique given in this study, the degraded image is restorable. Models III and IV are better than the existing approaches.

Figure 6 shows the effect of the initial threshold θ_0 on accuracy in PSNR P for each of initial neighborhood $N_0 = 1, 2, 3, 4, 5$ for models III and IV. In this case, P yields the maximum when $N_0 = 3$ and $\theta_0 = 96$ for model III and when $N_0 = 2$ and $\theta_0 = 98$ for model IV. Figure 5 (e) and (f) were restored by these values.

As an example of another image, Fig. 7 (a) shows the degraded image. As well as the above-mentioned image, the degraded image contains 30% noise compared to the original image. The condition of the computation is equal to that of the earlier description. According to the present algorithm, results of IRS, model I, model II, model III, and model IV are shown in Fig. 7 (b), (c), (d), (e), and (f), respectively. The initial neighborhood and the initial threshold are $N_0 = 3$ and $\theta_0 = 118$ for IRS, $N_0 = 7$ and $\theta_0 = 86$ for model I, $N_0 = 7$ and $\theta_0 = 85$ for model II, $N_0 = 2$ and $\theta_0 = 119$ for model III, and $N_0 = 2$ and $\theta_0 = 121$ for model IV, respectively. It is proven that the degraded image can be also restored in this case. Models III and IV are also greater than the existing approaches.

Figure 8 presents the effect of the initial threshold θ_0 on accuracy in PSNR P for each of initial neighborhood $N_0 = 1, 2, 3, 4, 5$ for models III and IV. In this case, P yields the maximum when $N_0 = 2$ and $\theta_0 = 119$ for model III and when $N_0 = 2$ and $\theta_0 = 121$ for model IV. Figure 7 (e) and (f) were restored by these values.

Table 2 summarizes PSNR for results of models I, II, III, and IV compared to the moving average filter (MAF), the median filter (MF), and conventional model (IRS).



(a) Degraded image i

(b) IRS

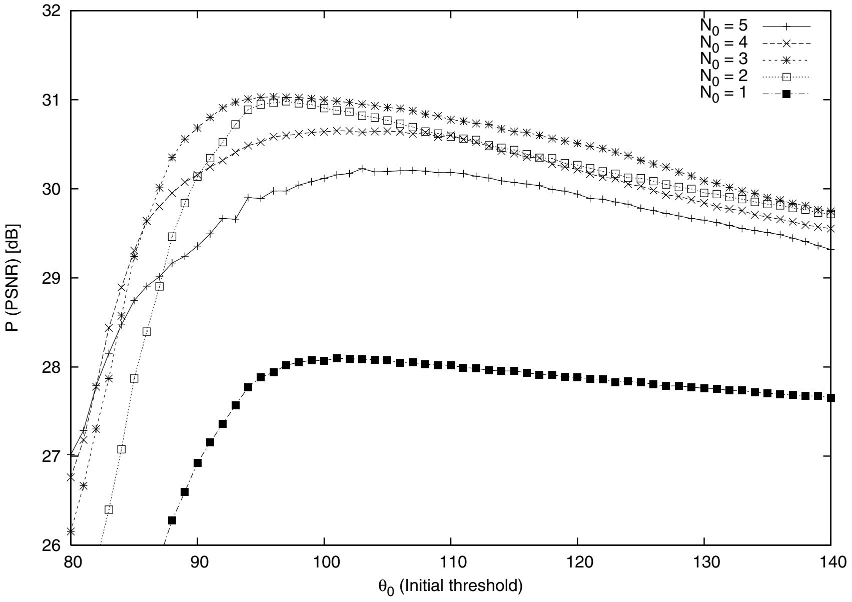
(c) Model I

(d) Model II

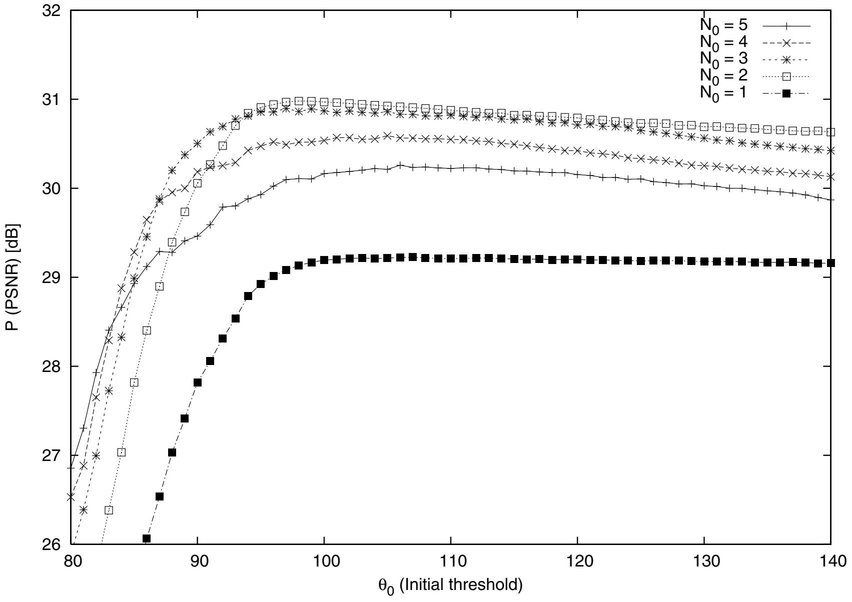
(e) Model III

(f) Model IV

Fig. 5. Degraded image i with 512×512 size and 256 gray-scale, and results of IRS, model I, model II, model III, and model IV

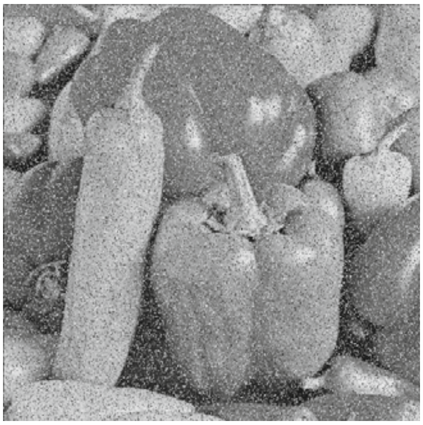


(a) Model III

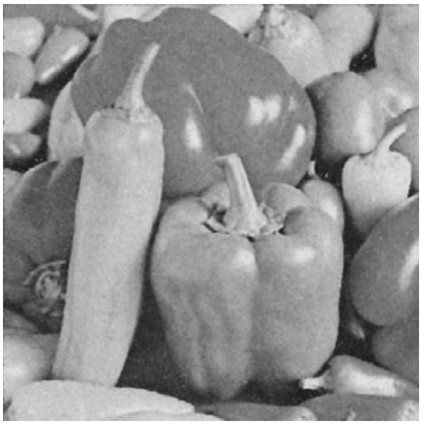


(b) Model IV

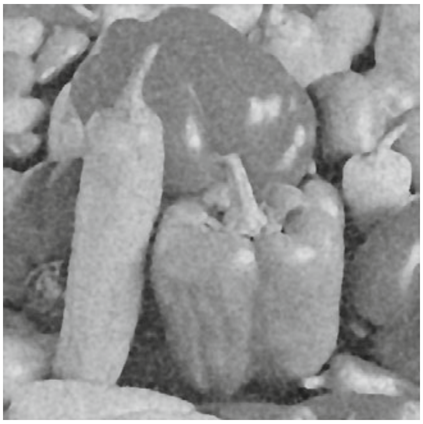
Fig. 6. PSNR and initial threshold for image i



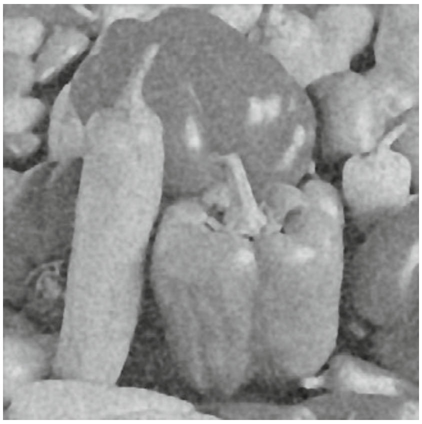
(a) Degraded image ii



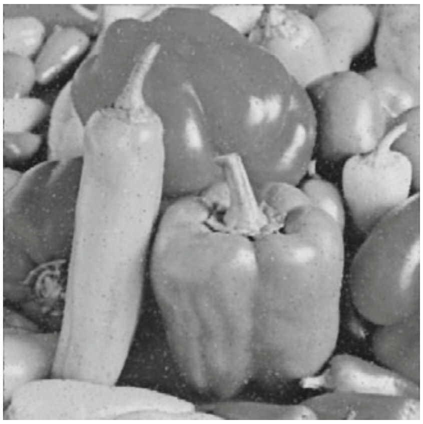
(b) IRS



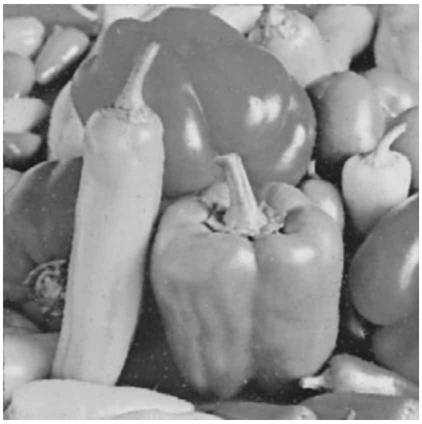
(c) Model I



(d) Model II

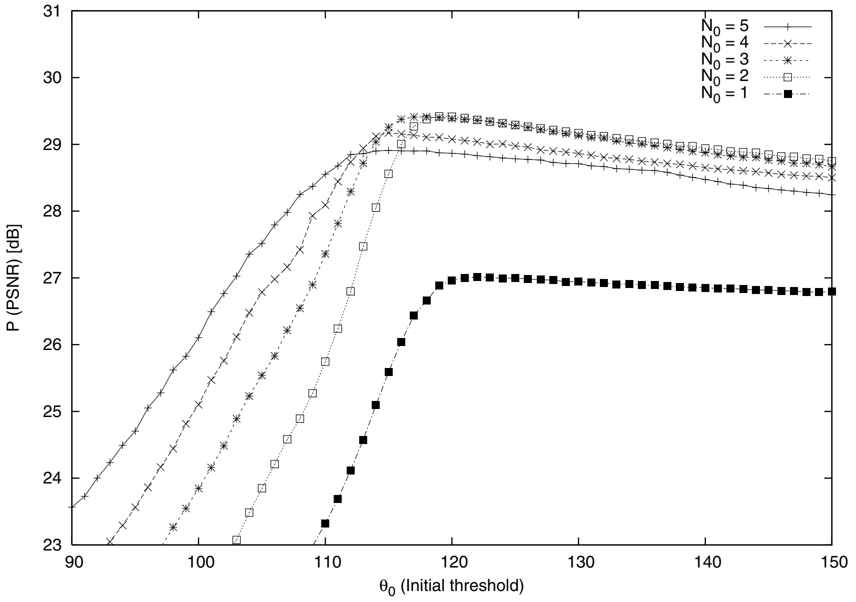


(e) Model III

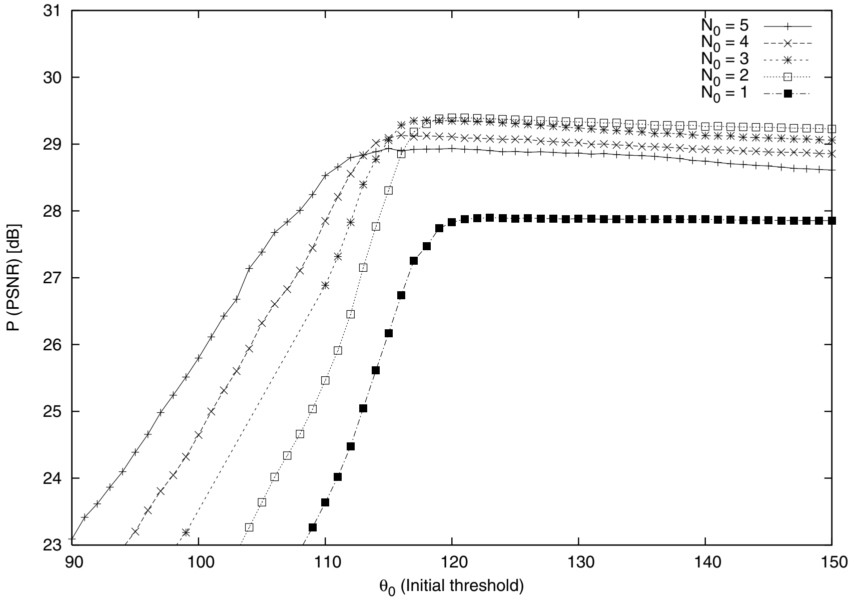


(f) Model IV

Fig. 7. Degraded image ii with 512×512 size and 256 gray-scale, and results of IRS, model I, model II, model III, and model IV



(a) Model III



(b) Model IV

Fig. 8. PSNR and initial threshold for image ii

The size of the filter mask in the smoothing methods (MAF and MF) is 3×3. It is proven that models III and IV excel MAF, MF, IRS, model I, and model II for both images i and ii.

For models III and IV, learning proceeds by receiving input features since neighboring pixels as well as a notice pixel are utilized. In the computational effect, the processing of model III is faster than that of model IV because the object pixels of computation are four and nine for model III and model IV, respectively.

5 Conclusions

In this study, algorithms of image restoration in self-organizing maps grounded on learning with neighboring inputs have been described and their validity has been shown through numerical experiments. Novel approaches were presented that neighboring pixels as well as a notice pixel are prepared as an input, and a degraded image was restored according to an algorithm of self-organizing maps. Our model formed a map in which one element corresponds to each pixel. Image inference was conducted by self-organizing maps using pixel values as input. A renewal function with threshold was introduced in proportion to the difference between input value and inferred value. Based on this function, our approach did not respond to input including noise oversensitively. As any pixel was influenced by neighboring pixels corresponding to neighboring setting, the inference of an original image was appropriately promoted. Finally, for the future works, we will study more effective techniques of our algorithms.

References

1. Grossberg, S.: Adaptive Pattern Classification and Universal Recoding: I. Parallel Development and Coding of Neural Feature Detectors. Biol. Cybern. 23, 121–134 (1976)
2. Willshaw, D.J., Malsburg, C.: How Patterned Neural Connections Can Be Set up by Self-Organization. Proc. R. Soc. Lond. B 194, 431–445 (1976)
3. Hertz, J., Krogh, A., Palmer, R.G.: Introduction to the Theory of Neural Computation. Addison-Wesley, Reading (1991)
4. Kohonen, T.: Self-Organizing Maps. Springer, Berlin (1995)
5. Villmann, T., Herrmann, M., Martinetz, T.M.: Topology Preservation in Self-Organizing Feature Maps: Exact Definition and Measurement. IEEE Trans. Neural Networks 8, 256–266 (1997)
6. Maeda, M., Miyajima, H., Shigei, N.: Parallel Learning Model and Topological Measurement for Self-Organizing Maps. Journal of Advanced Computational Intelligence and Intelligent Informatics 11, 327–334 (2007)
7. Durbin, R., Willshaw, D.: An Analogue Approach to the Traveling Salesman Problem Using an Elastic Net Method. Nature 326, 689–691 (1987)
8. Angéniol, B., Vaubois, G., Texier, J.-Y.: Self-Organizing Feature Maps and the Traveling Salesman Problem. Neural Networks 1, 289–293 (1988)
9. Ritter, H., Schulten, K.: On the Stationary State of Kohonen's Self-Organizing Sensory Mapping. Biol. Cybern. 54, 99–106 (1986)
10. Ritter, H., Schulten, K.: Convergence Properties of Kohonen's Topology Conserving Maps, Fluctuations, Stability, and Dimension Selection. Biol. Cybern. 60, 59–71 (1988)

11. Maeda, M., Miyajima, H.: Competitive Learning Methods with Refractory and Creative Approaches. IEICE Trans. Fundamentals E82–A, 1825–1833 (1999)
12. Maeda, M., Shigei, N., Miyajima, H.: Adaptive Vector Quantization with Creation and Reduction Grounded in the Equinumber Principle. Journal of Advanced Computational Intelligence and Intelligent Informatics 9, 599–606 (2005)
13. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Prentice-Hall, Englewood Cliffs (2002)
14. Geman, S., Geman, D.: Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. IEEE Trans. Pattern Anal. Mach. Intel. 6, 721–741 (1984)
15. Maeda, M., Miyajima, H.: State Sharing Methods in Statistical Fluctuation for Image Restoration. IEICE Trans. Fundamentals E87-A, 2347–2354 (2004)
16. Maeda, M.: A Relaxation Algorithm Influenced by Self-Organizing Maps. In: Kaynak, O., Alpaydın, E., Oja, E., Xu, L. (eds.) ICANN 2003 and ICONIP 2003. LNCS, vol. 2714, pp. 546–553. Springer, Heidelberg (2003)
17. Maeda, M., Shigei, N., Miyajima, H.: Learning Model in Relaxation Algorithm Influenced by Self-Organizing Maps for Image Restoration. IEE J. Trans. Electrical and Electronic Engineering 3, 404–412 (2008)
18. Gersho, A., Gray, R.M.: Vector Quantization and Signal Compression. Kluwer Academic Publishers, Dordrecht (1992)

On the Projection of k-Valued Non-linearly Separable Problems into m-Valued Linearly Separable Problems

Igor Aizenberg

Department of Computer Science, Texas A&M University-Texarkana
P.O. Box 5518, Texarkana, TX 75505-5518, U.S.A.
igor.aizenberg@tamut.edu

Abstract. In this paper, we observe a new approach to learn non-linearly separable problems using a single multi-valued neuron. It is shown that a *k*-valued problem, which is non-linearly separable in the *n*-dimensional space can be projected into an *m*-valued (where *m=kl*) linearly separable problem in the same space. This projection can be utilized through a periodic activation function for the multi-valued neuron. Then the initial problem can be learned by a single multi-valued neuron using its learning algorithm. This approach is illustrated by the examples of such problems as XOR, Parity *n*, mod *k* addition of *n* variables and some benchmarks using a single multi-valued neuron.

Keywords: Complex-valued neural networks, Multi-valued neuron, Derivative-free learning, XOR, Parity *n*, Mod *k* addition.

1 Introduction

It is commonly known that a linearly separable (threshold) Boolean function and a linearly separable function whose range is Boolean can be learned by a single neuron that is by the classical perceptron – a neuron with a threshold activation function 1]. It is also known that a linearly separable (threshold) multiple-valued function can also be learned by a single neuron – the multi-valued neuron 2].

However, it is commonly known that, for example, almost all real world classification problems are non-linearly separable, that is they are described by non-linearly separable (non-threshold) functions. Suppose we have to solve some *n*-dimansional *k*-class classification peoblem and this problem is described by some non-threshold function, which is either a discrete function of *k*-valued logic or a function with a descete *k*-valued range. The commonly used approach for solving such a problem is its consideration in the larger dimensional space. Actually, one of the ways to utilize this approach is a feedforward neural network, where hidden neurons form a new space, and a problem becomes linearly separable and solvable 1]. Another popular approach for solving this problem is the support vector machine (SVM) introduced in 3], where a larger dimensional space is formed using the support vectors and a problem becomes linearly separable and solvable in this new space.

In this paper, we will consider how the same problem can be approached from a different side. Suppose we have some *n*-dimansional *k*-valued non-linearly separable problem. This means that this problem is described by some non-threshold *k*-valued function of *n* variables. We will discover here how this non-threshold *k*-valued function of *n* variables can be projected into an *m*-valued partually defined threshold function of the same *n* variables (where $m > k$). This means that our *n*-dimansional *k*-valued non-linearly separable problem will become an *n*-dimansional *m*-valued linearly separable problem and therefore it will be possible to learn it using a single multi-valued neuron. Thus, we will reach a linear separation without extension of that initial *n*-dimansional space where our problem is defined.

A main tool, which will be used here to reach the declared goals, is the multi-valued neuron (MVN). The MVN was introduced in 4]. This neuron is based on the concept of multiple-valued logic over the field of complex numbers. This concept was introduced in 5], then presented in detail in 6], and further developed in 7]. The continuous MVN was introduced in 8] and then developed in 9]. A single MVN has significantly higher functionality than a single sigmoidal neuron. Another advantage of the MVN is its learning algorithm, which is derivative-free. The MVN's input/output mapping is always described by some threshold function of *k*-valued logic.

Another important tool, which will be used in this paper is the universal binary neuron (UBN). It was introduced in 10], then its theory was presented in detail in 2], and its modified learning algorithm is considered in 11]. The UBN is a Boolean neuron, its inputs and output are binary, thus it implements those input/output mappings, which are described by Boolean functions. The most important advantage of the UBN over traditional neurons is its ability to learn non-linearly separable Boolean functions. This is possible because the UBN employs complex-valued weights and the original activation function.

In this paper, we use the idea, which is behind the UBN activation function, to modify the MVN activation function. This idea is the periodicity of an activation function. A periodic activation function, which is used in the UBN makes it possible to project the Boolean (2-valued) logic into an *m*-valued logic, where $m > 2$. This makes it possible to project any non-linearly separable Boolean function of *n* variables into a partially defined (only on Boolean inputs) linearly-separable function of the same *n* variables. This approach can be generalized for non-linearly separable functions of *k*-valued logic for $k > 2$, and such functions can be projected into partially defined functions of *m*-valued logic, where $m > k$. This paper extends the author's recent work 12], where some preliminary results were reported.

2 Multi-Valued Neuron (MVN)

2.1 Discrete and Continuous MVN

The discrete MVN was proposed in 4]. It is based on the principles of multiple-valued threshold logic over the field of complex numbers. The discrete MVN performs a mapping between *n* inputs and a single output. This mapping is described by a multiple-valued (*k*-valued) threshold function of *n* variables $f(x_1, ..., x_n)$. It is important to specify that we consider here multiple-valued logic over the field of complex numbers 2], 6]. While in traditional multiple-valued logic its values are encoded by the integers from the set $K = \{0,1,...,k-1\}$, in the one over the field of complex numbers they are encoded by the *k*th roots of unity $E_k = \{\varepsilon^0, \varepsilon, \varepsilon^2, ..., \varepsilon^{k-1}\}$, where $\varepsilon^j = e^{i2\pi j/k}$, $j = 0,...,k-1$, (*i* is an imaginary unity). A *k*-valued threshold function describing the MVN input/output mapping, can be represented using *n*+ 1 complex-valued weight as follows

$$f(x_1, ..., x_n) = P(w_0 + w_1x_1 + ... + w_nx_n)\,, \tag{1}$$

where $x_1, ..., x_n$ are the variables (neuron inputs) and $w_0, w_1, ..., w_n$ are the weights. The values of the function and of the neuron inputs are the *k*th roots of unity: $\varepsilon^j = e^{i2\pi j/k}$, $j \in \{0,1,...,k-1\}$, *i* is an imaginary unity. *P* is the activation function

$$P(z) = e^{i2\pi j/k}, \text{ if } 2\pi j/k \le \arg z < 2\pi(j+1)/k, \tag{2}$$

where *j*=0, 1, ..., *k*-1 are values of *k*-valued logic, $z = w_0 + w_1x_1 + ... + w_nx_n$ is the weighted sum, arg *z* is the argument of the complex number *z*. It is important to mention that function (2), which was introduced in 5], is historically the first known complex-valued activation function. Function (2) divides a complex plane into *k* equal sectors and maps the whole complex plane into a set of *k*th roots of unity (see Fig. 1).

This approach was later generalized for the continuous case. The continuous MVN has been proposed in 8] and then developed in 9]. The continuous case corresponds to $k \to \infty$ in (2). If the number of sectors $k \to \infty$ (see Fig. 1), then the angular size of

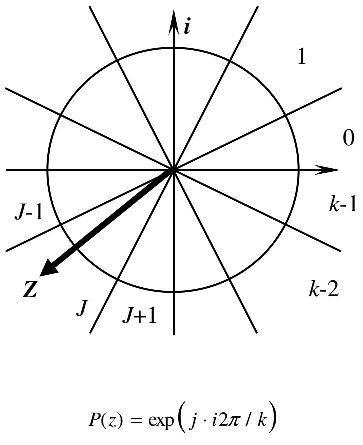


Fig. 1. Geometrical interpretation of the discrete MVN activation function.

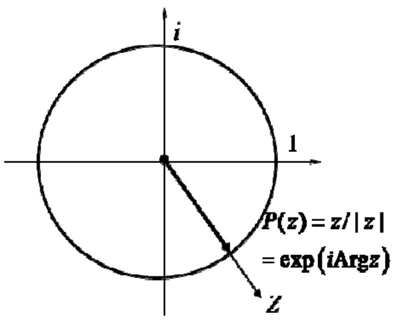


Fig. 2. Geometrical interpretation of the continuous MVN activation function.

a sector tends to 0. Hence, an activation function in this case becomes simply a projection of the weighted sum $z = w_0 + w_1x_1 + ... + w_nx_n$ onto the unit circle:

$$P(z) = \exp(i\ (\arg z)) = e^{iArg\ z} = z / |z|, \tag{3}$$

where z is the weighted sum, Arg z is a main value of its argument (phase) and $|z|$ is the absolute value of the complex number z.

Activation function (3) is illustrated in Fig. 2. It maps the whole complex plane into the unit circle. Evidently, a hybrid MVN (with continuous inputs/discrete output, discrete inputs/continuous output, hybrid inputs and discrete or continuous output) can also be easily considered.

2.2 MVN Learning

MVN learning algorithm is identical for both discrete and continuous neurons. The most important property of MVN learning is that it is derivative-free. There are two MVN learning algorithms. One of them, which was proposed in 2], is computationally more efficient. It is based on the error-correction learning rule. If T is the desired neuron's output and Y is the actual one, then $\delta = T - Y$ is the error (see Fig. 3) that determines that adjustment of the weights, which is performed as follows:

$$W_{r+1} = W_r + \frac{C_r}{(n+1)}\delta\overline{X}, \tag{4}$$

or, as it was suggested in 9], as follows

$$W_{r+1} = W_r + \frac{C_r}{(n+1)|z_r|}\delta\overline{X}, \tag{5}$$

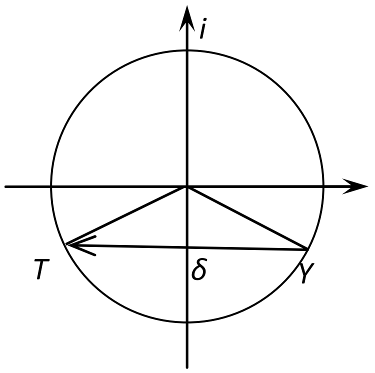


Fig. 3. Geometrical interpretation of the MVN learning rule

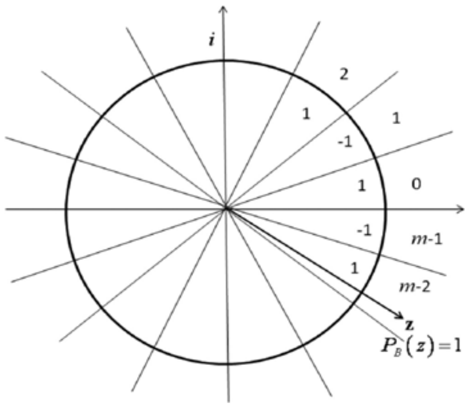


Fig. 4. Geometrical interpretation of the UBN activation function

where $\overline{X}$ is the vector of inputs with the components complex-conjugated, *n* is the number of neuron inputs, δ is the neuron's error, *r* is the number of the learning iteration, W_r is the current weighting vector (to be corrected), W_{r+1} is the following weighting vector (after correction), C_r is the constant part of the learning rate (it may always be equal to 1), and $|z_r|$ is the absolute value of the weighted sum obtained on the r^{th} iteration. A factor $1/|z_r|$ in (5) is a variable part of the learning rate. The use of it can be important for learning highly nonlinear input/output mappings.

3 Universal Binary Neuron

The UBN was introduced in 10]. It is presented in detail in 2], and its modified learning algorithm was recently considered in 11]. A key idea behind the UBN is the use of complex-valued weights and an original activation function for learning non-linearly separable Boolean functions.

If *k*=2 in (2) then the complex domain is divided into two parts (the top semiplane ("1") and the bottom semiplane ("-1")):

$$P(z) = \begin{cases} 1, & 0 \le \arg(z) < \pi \\ -1, & \pi \le \arg(z) < 2\pi. \end{cases}$$

However, this activation function does not increase the neuron's functionality: although the weights are complex, the neuron still can only learn linearly separable Boolean functions [2]. It was suggested in 11] to use the following periodic activation function

$$P_B(z) = (-1)^j, \text{if } \; 2\pi j/m \le \arg(z) < 2\pi(j+1)/m; \; m = 2l, l \in \mathbb{N}, \tag{6}$$

where *l* is some positive integer, *j* is a non-negative integer $0 \le j < m$. Activation function (6) is illustrated in Fig. 4. It separates the complex plane into $m = 2l$ equal sectors and determines the neuron's output by the alternating periodic sequence of 1, -1, 1, -1,...1, -1 depending on the parity of the sector's number.

Activation function (6) determines the most important advantage of the UBN. It is its ability to learn non-linearly separable Boolean functions.

For example, the favorite non-linearly separable problems of the research community - XOR and Parity *n* become linearly separable and can be learned by a single UBN with activation function (6). Table 1 shows how the XOR problem can be solved using a single UBN with the activation function (6) ($l = 2, m = 4$).

The UBN learning algorithm is presented in detail in 11]. It is based on the same learning rules (4) and (5) as the MVN learning algorithm. The choice of the desired output is based on the closeness of the current weighted sum to the right or left adjacent sector.

Table 1. Solving the **XOR** problem using a single UBN with the weighting vector (0, *i*, 1)

#	x_1	x_2	$z = w_0 + w_1 x_1 + w_2 x_2$	$P_B(z)$	XOR= $= x_1 \underset{\text{mod} 2}{\oplus} x_2$
1)	**1**	**1**	1 + *i*	**1**	**1**
2)	**1**	**-1**	-1 + *i*	**-1**	**-1**
3)	**-1**	**1**	1 - *i*	**-1**	**-1**
4)	**-1**	**-1**	-1 - *i*	**1**	**1**

Let us analyze activation function (6). It is a periodic function with the period 2. Comparing this function with (2), we may conclude that (6) projects the Boolean logic into an *m*-valued logic. Activation function (6) may also be written in the following form:

$$P_B(z) = j \bmod 2, \text{ if } 2\pi j / m \le \arg z < 2\pi (j+1)/m, \\ j = 0,1,...,m-1; \; m = 2l, l \ge 2. \tag{7}$$

It was distinguished in 11] that if some non-threshold Boolean function can be learned using a single UBN with activation function (6) and its weighting vector $(w_0, w_1, ..., w_n)$ results from the learning process, then there exists a partially defined (on the Boolean variables) *m*-valued threshold function, which can be implemented by a single MVN with the same weighting vector. This means that a non-threshold Boolean function, which cannot be learned using a traditional perceptron, can be projected to a partially defined *m*-valued threshold function, which can be learned using a single MVN. Let us consider how the same approach can be used to learn using a single neuron not only non-linearly separable Boolean functions, but also non-linearly separable multiple-valued functions.

4 A Periodic Activation Function for the MVN

Let $E_k = \{1, \varepsilon_k, \varepsilon_k^2, ..., \varepsilon_k^{k-1}\}$ (where $\varepsilon_k = e^{i2\pi/k}$ is the primitive *k*th root of unity) be the set of the *k*th roots of unity. Let *O* be the continuous set of the points located on the unit circle. Let $K = \{0, 1, ..., k-1\}$ be the set of the values of *k*-valued logic. Let $f(x_1, ..., x_n)$ be a function and either $f : E_k^n \to K$ or $f : O^n \to K$. Hence, the range of f is discrete, while its domain is either discrete or continuous. In general, its domain may be even hybrid. If some function $f(y_1, ..., y_n), y_i \in [a_j, b_j], a_j, b_j \in \mathbb{R}, j = 1, ..., n$ is defined on the bounded subdomain $D^n \subset \mathbb{R}^n$ ($f : D^n \to K$), then it can be easily transformed to $f : O^n \to K$ by a simple linear transformation applied to each variable:

$y_j \in [a_j, b_j] \Rightarrow \varphi_j = \frac{y_j - a_j}{b_j - a_j}\alpha \in [0, 2\pi],$ and then $x_j = e^{i\varphi_j} \in O, j = 1, 2, ..., n$ is the

$j = 1, ..., n; 0 < \alpha < 2\pi,$

complex number located on the unit circle. Hence, we obtain the function $f(x_1, ..., x_n): O^n \to K$.

If this function $f(x_1, ..., x_n)$ is not a *k*-valued threshold function, it cannot be learned by a single MVN with activation function (2).

Let us project the *k*-valued function $f(x_1, ..., x_n)$ into an *m*-valued logic, where $m = kl$, l is integer and $l \geq 2$. To do this, let us define the following new discrete activation function for the MVN:

$$\begin{aligned} &P_l(z) = j \bmod k, \text{ if } 2\pi j / m \leq \arg z < 2\pi (j+1) / m, \\ &j = 0, 1, ..., m-1; \; m = kl, l \geq 2. \end{aligned} \tag{8}$$

This definition is illustrated in Fig. 5. Activation function (8) separates the complex plane onto m equal sectors and $\forall t \in K$ there are exactly l sectors, where (8) equals to t.

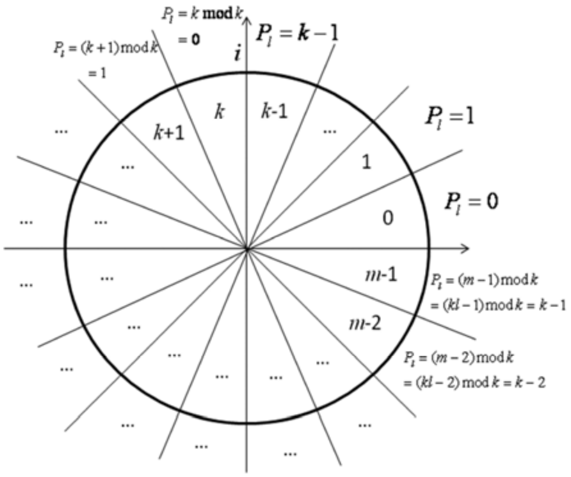

$P_l = k \bmod k$ $= 0$
$P_l = k-1$
$P_l = (k+1) \bmod k$ $= 1$
i
k
k-1
k+1
$P_l = 1$
1
$P_l = 0$
0
m-1
m-2
$P_l = (m-1) \bmod k$ $= (kl-1) \bmod k = k-1$
$P_l = (m-2) \bmod k$ $= (kl-2) \bmod k = k-2$
...

Fig. 5. Geometrical interpretation of the *l*-repeated multiple discrete-valued MVN activation function (7)

This means that activation function (8) establishes mappings from E_k into $E_m = \{1, \varepsilon_m, \varepsilon_m^2, ..., \varepsilon_m^{k-1}\}$ and from K into $M = \{0, 1, ..., k-1, k, k+1, ..., m-1\}$, respectively. Since $m = kl$ then each element from M and E_m has exactly l prototypes in K and E_k, respectively. In turn, this means that the MVN's output, depending on which one of the m sectors (whose ordinal numbers are determined by the elements of the set M) the weighted sum is located in, is equal to

$$\underbrace{\underbrace{0,1,...,k-1}_{0},\underbrace{0,1,...,k-1}_{1},...,\underbrace{0,1,...,k-1}_{l-1}}_{lk=m}. \tag{9}$$

Hence the MVN's activation function in this case becomes k-periodic.

Activation function (8) projects a k-valued function $f(x_1,...,x_n)$ into m-valued logic. This projection becomes very attractive, if $f(x_1,...,x_n)$, being a non-threshold function in k-valued logic, becomes a partially defined threshold function in m-valued logic. The latter means that $f(x_1,...,x_n)$ can be learned it using a single MVN. It will be shown below that this is definitely the case.

It is important to mention that if $l=1$ in (8) then $m=k$ and activation function (8) coincides with activation function (2) accurate within the interpretation of the neuron's output (if the weighted sum is located in the jth sector then according to (2) the neuron's output is equal to $e^{ij2\pi/k}=\varepsilon^j \in E_k$, which is the jth kth root of unity, while in (8) it is equal to $j \in K$). Evidently, the multiple-valued activation function (8) is a generalization of the binary activation function (7) for the multiple-valued case.

5 Learning Algorithm for the MVN with a Periodic Activation Function

To make the approach proposed in Section 3 active, it is necessary to develop an efficient learning algorithm for the MVN with activation function (8). Such an algorithm will be presented here.

As it was mentioned above (Section 2), one of the MVN learning algorithms is based on either of error-correction learning rules (4) or (5). Let us adapt this algorithm to activation function (8).

Let $f(x_1,...,x_n)$ be a function of k-valued logic and $f: E_k^n \to K$ or $f: O^n \to K$. Let this function be non-threshold and therefore it is not possible to learn it using a single MVN with activation function (2). Let us try to learn it in m-valued logic using a single MVN with activation function (8). Thus, the expected result of this learning process is the representation of $f(x_1,...,x_n)$ according to (1), where the activation function P_l determined by (8) substitutes for the activation function P determined by (2).

This learning process may be based on either of the same learning rules (4) or (5), but applied to $f(x_1,...,x_n)$ as to the m-valued function. To use these learning rules, it is necessary to determine a desired output. Unlike the case of the MVN with activation function (2), a desired output in terms of m-valued logic cannot be determined unambiguously for the MVN with activation function (8). According to

(8), there are exactly l sectors on the complex plane out of m, where this activation function equals to the given desired output $t \in K$. Therefore, there are exactly l out of mth roots of unity that can be used as the desired outputs in (4) or (5). Which one of them should we choose? We suggest using the same approach that was used in the error-correction learning algorithm for the UBN 2], 11]. UBN's activation function (6) determines an alterning sequence with respect to sectors on the complex plane. Hence, if the actual output of the UBN is not correct, in order to make the correction, we can "move" the weighted sum into either of the sectors adjacent to the one where the current weighted sum is located. It was suggested to always move it to the sector, which is closest to the current weighted sum (in terms of the angular distance). The convergence of this learning algorithm was proven in 2].

Let us use the same approach here. Activation function (8) determines k-periodic sequence (9) with respect to sectors on the complex plane. Suppose that the current MVN's output is not correct and the current weighted sum is located in the sector $s \in M = \{0,1,...,m-1\}$. Since $l \geq 2$ in (8), there are l sectors on the complex plane, where function (8) takes the correct value. Two of these l sectors are the closest ones to sector s (from right and left sides, respectively). From these two sectors, we choose sector q whose border is closest to the current weighted sum z. Then either of learning rules (4) or (5) can be applied. Hence, the learning algorithm for the MVN with activation function (8) can be described as follows. Let us have N learning samples for the function $f(x_1,...,x_n)$ to be learned and $j \in \{1,...,N\}$ be the number of the current learning sample (initially, $j=1$). One iteration of the learning process consists of the following steps:

1) Set r=1, j=1, and Learning="False".
2) Check (1) with the activation function (8) for the learning sample j.
3) If (1) holds then set $j = j+1$, otherwise set Learning='True' and go to Step 5.
4) If $j \leq N$ then go to Step 2, otherwise go to Step 9.
5) Let z be the current value of the weighted sum and $P(z) = \varepsilon^s, s \in M$, $P(z)$ is the activation function (2), where m is substituted for k. Hence the MVN's actual output is $P_l(z) = s \bmod k$. Find $q_1 \in M$, which determines the closest sector to the sth one, where the output is correct, from the right, and find $q_2 \in M$, which determines the closest sector to the sth one, where the output is correct, from the left (this means that $q_1 \bmod k = d$ and $q_2 \bmod k = d$).
6) If $\left(\arg z - \arg\left(e^{i(q_1+1)2\pi/m}\right)\right) \bmod 2\pi \leq \left(\arg\left(e^{iq_2 2\pi/m}\right) - \arg z\right) \bmod 2\pi$ then $q = q_1$ else $q = q_2$.
7) Apply the learning rule (4) or (5) to adjust the weights.
8) Set $j = j+1$ and return to Step 4.

9) If Learning=‘False’ then go to Step 10, otherwise set *r*=*r*+1, *j*=1, Learning=’False’ and go to Step 2.

10) End.

Since according to this learning algorithm the learning of a *k*-valued function is reduced to the learning of a partially defined *m*-valued function, the convergence of the learning algorithm may be proven in the similar manner as the convergence of the learning algorithm based on rule (4) and of the UBN learning algorithm were proven in 2].

6 Simulation Results

To confirm the efficiency of the proposed activation function and learning algorithm, they were checked for the following three problems using a software simulator written in Borland Delphi 5.0 running on a PC with the Intel® Core™2 Duo CPU.

6.1 Wisconsin Breast Cancer (Diagnostic)

This famous benchmark database was downloaded from the UC Irvine Machine Learning Repository 13]. It contains 569 samples that are described by 30 real-valued features. There are two classes (“malignant” and “benign”) to which these samples belong.

A single MVN with activation function (2) with $k=2$ fails to learn the entire data set because it is non-linearly separable. However, a single MVN with activation function (8) with $k=2, l=2, m=4$ learns the problem with no errors. Ten runs of the learning process started from different random weights resulted in convergence after 649-1300 iterations, which took a few seconds.

We have also tested the ability of a single MVN to solve classification problem. 10-fold cross validation was used as it is recommended in 13]. Every time the data set was divided into a learning set (400 samples) and a testing set (169) samples. After the learning set was learned completely with no errors, the classification of the testing set samples was performed. A classification rate of 96.5%-97.5% was achieved. This is comparative to the best known result (97.5%) 13]. However, it is important to note that our method solves the problem using just a single neuron, while in the previous works either different networks or linear programming methods were used.

6.2 Sonar

This famous benchmark database was also downloaded from the UC Irvine Machine Learning Repository 13]. It contains 208 samples that are described by 60 real-valued features. There are two classes (“mine” and “rock”) to which these samples belong.

A single MVN with activation function (2) with $k=2$ fails to learn the entire data set even after 1,000,000 iterations. However, the single MVN with activation function (8) with $k=2, l=2, m=4$ learns the problem with no errors. Ten runs of the learning process started from different random weights resulted in convergence after 65-180 iterations, which took a few seconds.

We have also tested the ability of a single MVN to solve the classification problem. This set is initially divided by its creators into learning (104 samples) and testing (104 samples) subsets. After the learning set was learned completely with no errors, the classification of the testing set samples was performed. The classification rate of 88.1%-91.5% was achieved. This is comparative to the best known results reported in 14] – 94% (Fuzzy Kernel Perceptron), 89.5% (SVM), and in 9] - 88%-93% (MLMVN). However, here the problem was solved using just a single neuron, while in the previous works different neural and kernel-based networks were used.

6.3 k-Valued Non-threshold Function

Let us consider the following fully defined function of 3 variables of 4-valued logic $f(x_1, x_2, x_3) = (x_1 + x_2 + x_3) \bmod 4$ (see the first four columns of Table 2). This function, which is the analogue of parity 3 function in the Boolean logic, is non-threshold in 4-valued logic and cannot be learned using a single MVN with activation function (2) with $k = 4$. However, this function can be learned by a single MVN with activation function (8) with $k = 4, l = 8, m = 32$ (see columns 5-6 of Table 2). The learning process converges after 584-43875 iterations (ten independent runs).

It is interesting that every time the learning process has converged to different weighting vectors, but to the same type of a resulting monotonic *m*-valued function (see the 5th column of the Table 2). This confirms that the learning of a non-threshold *k*-valued function may be reduced to the learning of a partially defined *m*-valued threshold function.

7 Conclusions

We have considered in this paper the original approach for projecting a *k*-valued non-linearly separable function of *n* variables to a partially defined *m*-valued threshold function of *n* variables. This makes it possible to learn such no-linearly separable multiple-valued functions using a single multi-valued neuron like it is possible to learn non-linearly separable Boolean functions using a single universal binary neuron. This means that non-linearly separable classification problems, which have been considered unsolvable using a single neuron, can now be learned using a single multi-valued neuron. The projection of a non-linearly separable problem to a linearly separable problem described in the paper is reached by using a new periodic activation function for the multi-valued neuron and by employing a modified learning algorithm for this neuron.

Acknowledgements. This work is supported by the National Science Foundation under Grant 0925080.

Table 2. Non-threshold function of 3 variables of 4-valued logic and the results of its learning

x_1	x_2	x_3	$f(x_1, x_2, x_3)$ 4-valued	$j \in M$, $M = \{0,1,\ldots,31\}$	$P_l =$ $= j \bmod 4$
0	0	0	0	8	0
0	0	1	1	9	1
0	0	2	2	10	2
0	0	3	3	11	3
0	1	0	1	9	1
0	1	1	2	10	2
0	1	2	3	11	3
0	1	3	0	12	0
0	2	0	2	10	2
0	2	1	3	11	3
0	2	2	0	12	0
0	2	3	1	13	1
0	3	0	3	11	3
0	3	1	0	12	0
0	3	2	1	13	1
0	3	3	2	14	2
1	0	0	1	9	1
1	0	1	2	10	2
1	0	2	3	11	3
1	0	3	0	12	0
1	1	0	2	10	2
1	1	1	3	11	3
1	1	2	0	12	0
1	1	3	1	13	1
1	2	0	3	11	3
1	2	1	0	12	0
1	2	2	1	13	1
1	2	3	2	14	2
1	3	0	0	12	0
1	3	1	1	13	1
1	3	2	2	14	2
1	3	3	3	15	3
2	0	0	2	10	2
2	0	1	3	11	3
2	0	2	0	12	0
2	0	3	1	13	1
2	1	0	3	11	3
2	1	1	0	12	0
2	1	2	1	13	1
2	1	3	2	14	2
2	2	0	0	12	0
2	2	1	1	13	1
2	2	2	2	14	2
2	2	3	3	15	3
2	3	0	1	13	1
2	3	1	2	14	2
2	3	2	3	15	3
2	3	3	0	16	0
3	0	0	3	11	3
3	0	1	0	12	0
3	0	2	1	13	1
3	0	3	2	14	2
3	1	0	0	12	0
3	1	1	1	13	1
3	1	2	2	14	2
3	1	3	3	15	3
3	2	0	1	13	1
3	2	1	2	14	2
3	2	2	3	15	3
3	2	3	0	16	0
3	3	0	2	14	2
3	3	1	3	15	3
3	3	2	0	16	0
3	3	3	1	17	1

References

1. Haykin, S.: Neural Networks and Learning Machines, 3rd edn. Prentice Hall, New Jersey (2009)
2. Aizenberg, I., Aizenberg, N., Vandewalle, J.: Multi-valued and universal binary neurons: theory, learning, applications. Kluwer Academic Publishers, Boston (2000)
3. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1995)
4. Aizenberg, N.N., Aizenberg, I.N.: CNN Based on Multi-Valued Neuron as a Model of Associative Memory for Gray-Scale Images. In: The Second IEEE Int. Workshop on Cellular Neural Networks and their Applications, pp. 36–41. Technical University Munich, Germany (1992)
5. Aizenberg, N.N., Ivaskiv, Y.L., Pospelov, D.A.: About one generalization of the threshold function Doklady Akademii Nauk SSSR (The Reports of the Academy of Sciences of the USSR), 196(6) 1287–1290 (1971) (in Russian)
6. Aizenberg, N.N., Ivaskiv, Y.L.: Multiple-Valued Threshold Logic. Naukova Dumka Publisher House, Kiev (1977) (in Russian)
7. Aizenberg, I., Aizenberg, N., Vandewalle, J.: Multi-valued and universal binary neurons: theory, learning, applications. Kluwer Academic Publishers, Boston (2000)
8. Aizenberg, I., Moraga, C., Paliy, D.: A Feedforward Neural Network based on Multi-Valued Neurons. In: Reusch, B. (ed.) Computational Intelligence, Theory and Applications. Advances in Soft Computing. AISC, vol. XIV, pp. 599–612. Springer, Heidelberg (2005)
9. Aizenberg, I., Moraga, C.: Multilayer Feedforward Neural Network Based on Multi-Valued Neurons (MLMVN) and a Backpropagation Learning Algorithm. Soft Computing 11(2), 169–183 (2007)
10. Aizenberg, I.N.: A Universal Logic Element over the Complex Field. Kibernetika (Cybernetics and Systems Analysis) 27(3), 116–121 (in Russian) ; English version is available from Springer 27(3), 467–473 (1991)
11. Aizenberg, I.: Solving the XOR and Parity n Problems Using a Single Universal Binary Neuron. Soft Computing 12(3), 215–222 (2008)
12. Aizenberg, I.: A Multi-Valued Neuron with a Periodic Activation Function. In: International Joint Conference on Computational Intelligence, Funchal-Madeira, Portugal, October 5-7, pp. 347–354 (2009)
13. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository. University of California, Irvine (2007), http://www.ics.uci.edu/~mlearn/MLRepository.html
14. Chen, J.-H., Chen, C.-S.: Fuzzy Kernel Perceptron. IEEE Transactions on Neural Networks 13, 1364–1373 (2002)

Dual Adaptive Neurocontrol of Mobile Robots Using the Unscented Transform: Monte Carlo and Experimental Validation

Marvin K. Bugeja* and Simon G. Fabri

Department of Systems and Control Engineering
University of Malta, Msida MSD2080, Malta
{mkbuge,sgfabr}@eng.um.edu.mt

Abstract. In contrast to most adaptive schemes, dual adaptive controllers do not rely on the heuristic certainty equivalence assumption, but aim to strike a balance between estimation and control at all times. Yet, few such controllers have ever been implemented and tested in practice, especially within the context of intelligent control, and to the best of our knowledge none on mobile robots. With the help of Mont Carlo simulation and real-life experiments, this article presents and validates a novel dual adaptive neurocontroller based on the unscented transform, for the dynamic control of nonholonomic wheeled mobile robots. The robot nonlinear dynamic functions are unknown to the controller and a multilayer perceptron neural network, trained via an unscented Kalman predictor, is used for their approximation in real-time. Moreover, the proposed novel dual adaptive control law employs the unscented transform to improve further the system's performance.

Keywords: Dual adaptive control, Neural networks, Unscented Kalman filter, Unscented transform, Mobile robots.

1 Introduction

Many works on the control of nonholonomic wheeled mobile robots (WMRs) completely ignore the robot dynamics and rely on the assumption that the control inputs instantaneously establish the desired wheel velocities [1,2]. Others, that explicitly account for the robot dynamics due to its mass, friction and inertia, show that dynamic control leads to an improvement in tracking performance [3,4]. However, as argued in [3], perfect knowledge of the robot dynamics is unavailable in practice. In addition, these parameters can also vary over time due to loading, wear and ground conditions. These issues inspired the development of several robust and adaptive WMR controllers over the last decade. These include: pre-trained artificial neural network (ANN) based controllers and robust sliding-mode methods [4], parametric adaptive schemes [5], and functional adaptive controllers [6].

However, all these adaptive controllers rely on the heuristic certainty equivalence (HCE) assumption. This means that the estimated parameters/functions are used by the

* This work was supported by National RTDI under Grant 2004 RTDI-2004-026.

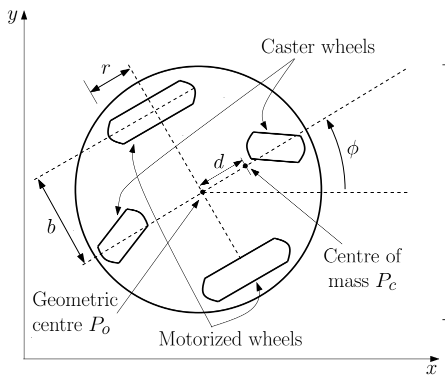
controller as if they were the true values, thereby ignoring completely the inherent uncertainty in the estimations. When the uncertainty is large, for instance during startup or when the dynamic parameters/functions vary, HCE control often leads to large tracking errors and excessive control actions, which can excite unmodelled dynamics or even lead to instability [7]. In contrast, the so-called *dual adaptive* controllers based on the *dual control* principle introduced by A. Fel'dbaum in the early 1960s [8], do not rely on the HCE assumption but account for the estimates' uncertainty in the control design. Specifically, a dual adaptive control law is designed with two aims in mind: (i) to ensure that the output tracks the reference signal with due consideration given to the estimates' uncertainty; (ii) to excite the plant input sufficiently so as to accelerate the estimation process, thereby reducing quickly the uncertainty in future estimates. These two features are known as *caution* and *probing* respectively [7,9].

Of the few dual adaptive controllers proposed, only our work presented in [10] focuses on the dynamic control of WMRs. However, the multilayer perceptron (MLP) dual adaptive scheme employed in this work, not only uses the extended Kalman filter (EKF) (which inherently involves a first-order Taylor approximation) as a neuro-estimator, but the control law itself is based on another first-order approximation of the measurement model. In contrast, the novelty of the control scheme presented in this article comprises: the use of a specifically devised form of the unscented Kalman filter (UKF) [11,12] as a recursive weight tuning algorithm instead of the EKF; and more importantly the development of a novel dual adaptive control law based on the unscented transform (UT) [11], instead of the first-order Taylor approximation. Together, these novel developments lead to a significant improvement in performance over the EKF-based scheme in [10]. To the best of our knowledge, this is the first time that the UT is being used in the context of dual adaptive control. Moreover, one should note that very few adaptive controllers have ever been implemented and tested on a physical WMR, amongst which one finds [13,5]. However, none of these address fully the uncertainty in the WMR dynamic functions nor take a dual adaptive control approach.

The rest of the article is organized as follows. Section 2 presents the mathematical model of the WMR. In Section 3 we present the novel UT-based dual adaptive dynamic control scheme. Simulation results, including those from a Monte Carlo comparative analysis, and experimental results are then presented in Section 4, which is followed by a brief conclusion in Section 5.

2 WMR Mathematical Model

Figure 1 depicts the differentially driven wheeled mobile robot considered in this article. The passive (caster) wheels are ignored, and the notation included in the figure is adopted throughout the article. The robot coordinate vector is denoted by $\boldsymbol{q} = [x\ y\ \phi\ \theta_r\ \theta_l]^T$, where (x, y) is the Cartesian coordinate of P_o, ϕ is the robot's orientation with reference to the x-axis, and θ_r, θ_l are the angular displacements of the right and left motorized wheels respectively. The *pose* of the robot refers to the vector $\boldsymbol{p} = [x\ y\ \phi]$.



P_o	midpoint on the driving axle
P_c	centre of mass of platform (no wheels)
d	distance from P_o to P_c
b	distance from each wheel to P_o
r	radius of each wheel
m_c	mass of the platform
m_w	mass of each wheel
I_c	angular mass of the platform about P_c
I_w	angular mass of each wheel about axle
I_m	angular mass of each wheel about diameter

Fig. 1. Differentially driven wheeled mobile robot

2.1 Kinematics

Assuming that the wheels roll without slipping, the kinematic model of this WMR, detailed in [10], is given by:

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \frac{r}{2}\cos\phi & \frac{r}{2}\cos\phi \\ \frac{r}{2}\sin\phi & \frac{r}{2}\sin\phi \\ \frac{r}{2b} & -\frac{r}{2b} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{\nu}, \tag{1}$$

where the wheel velocity vector $\boldsymbol{\nu} \triangleq [\nu_r \;\; \nu_l]^T = \left[\dot{\theta}_r \;\; \dot{\theta}_l\right]^T$.

2.2 Dynamics

The WMR dynamic model, also detailed in [10], is given by:

$$\bar{\boldsymbol{M}}\dot{\boldsymbol{\nu}} + \bar{\boldsymbol{V}}(\boldsymbol{\nu}) + \bar{\boldsymbol{F}}(\boldsymbol{\nu}) = \boldsymbol{\tau}, \tag{2}$$

where:

$$\bar{\boldsymbol{M}} = \begin{bmatrix} \frac{r^2}{4b^2}(mb^2 + I) + I_w & \frac{r^2}{4b^2}(mb^2 - I) \\ \frac{r^2}{4b^2}(mb^2 - I) & \frac{r^2}{4b^2}(mb^2 + I) + I_w \end{bmatrix}, \bar{\boldsymbol{V}}(\boldsymbol{\nu}) = \frac{m_c d r^3}{4b^2} \begin{bmatrix} \nu_r\nu_l - \nu_l^2 \\ \nu_r\nu_l - \nu_r^2 \end{bmatrix},$$

$m = m_c + 2m_w$, $I = (I_c + m_c d^2) + 2(I_m + m_w b^2)$, $\bar{\boldsymbol{F}}(\boldsymbol{\nu})$ is a vector of wheel velocity-dependent frictional terms, and $\boldsymbol{\tau} = [\tau_r \; \tau_l]^T$ with τ_r and τ_l being the torques applied to the right and left motorized wheels respectively.

To account for the fact that the controller is finally implemented on a digital computer, the continuous-time dynamics (2) are discretized through a first-order forward Euler approximation with a sampling interval of T seconds, resulting in

$$\boldsymbol{\nu}_k - \boldsymbol{\nu}_{k-1} = \boldsymbol{f}_{k-1} + \boldsymbol{G}_{k-1}\boldsymbol{\tau}_{k-1}, \tag{3}$$

where subscript k denotes that the corresponding variable is evaluated at kT seconds, and vector $\boldsymbol{f}_{k-1}$ and matrix $\boldsymbol{G}_{k-1}$, which together encapsulate the WMR nonlinear dynamics, are given by

$$\begin{aligned} \boldsymbol{f}_{k-1} &= -T\bar{\boldsymbol{M}}_{k-1}^{-1}\left(\bar{\boldsymbol{V}}_{k-1} + \bar{\boldsymbol{F}}_{k-1}\right), \\ \boldsymbol{G}_{k-1} &= T\bar{\boldsymbol{M}}_{k-1}^{-1}. \end{aligned} \tag{4}$$

The following conditions are assumed to hold:

Assumption 1. *The control input vector $\boldsymbol{\tau}$ remains constant over a sampling interval of T seconds (zero-order hold).*

Assumption 2. *The sampling interval T is chosen low enough for the Euler approximation error to be negligible.*

3 Control Design

The trajectory tracking task of a nonholonomic WMR is chosen as a test problem in this article. Nevertheless, the applicability of the dual-adaptive controller proposed in this section is much wider, and the resulting algorithm can be easily adapted for other tasks. In trajectory tracking the robot is required to track a non-stationary kinematically identical *virtual vehicle*, in *both* pose and velocity at all times, by minimizing the tracking error vector $\boldsymbol{e}_k$ defined in [1] as

$$\boldsymbol{e}_k = \begin{bmatrix} \cos\phi_k & \sin\phi_k & 0 \\ -\sin\phi_k & \cos\phi_k & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\boldsymbol{p}_{\boldsymbol{r}k} - \boldsymbol{p}_k\right), \tag{5}$$

where $\boldsymbol{p}_{\boldsymbol{r}k} = [x_{rk} \;\; y_{rk} \;\; \phi_{rk}]^T$ denotes the virtual vehicle pose vector. Consequently, the trajectory tracking control task is to make $\boldsymbol{e}_k$ converge to zero so that $\boldsymbol{p}_k$ converges to $\boldsymbol{p}_{\boldsymbol{r}k}$.

3.1 Kinematic Control

To address the trajectory tracking problem we employ a discrete-time version of the well-established kinematic control law originally proposed in [1], given by

$$\boldsymbol{\nu}_{\boldsymbol{c}k} = \begin{bmatrix} \frac{1}{r} & \frac{b}{r} \\ \frac{1}{r} & -\frac{b}{r} \end{bmatrix} \begin{bmatrix} v_{rk}\cos e_{3k} + k_1 e_{1k} \\ \omega_{rk} + k_2 v_{rk} e_{2k} + k_3 v_{rk}\sin e_{3k} \end{bmatrix},$$

where $\boldsymbol{\nu}_{\boldsymbol{c}k}$ is the wheel velocity-command vector computed by the kinematic controller, k_1, k_2, and k_3 are *positive* design parameters, v_{rk} and ω_{rk} are the translational and angular *virtual vehicle* velocities respectively (assumed to be *continuous* functions, at least know one sampling interval ahead), and e_{1k}, e_{2k}, e_{3k} are the elements of $\boldsymbol{e}_k$ in (5).

If one assumes *perfect velocity tracking* (*i.e.* $\boldsymbol{\nu}_k = \boldsymbol{\nu}_{\boldsymbol{c}k} \;\; \forall \;\; k$), hence ignoring the WMR dynamics expressed in (2), then this kinematic control law alone solves the trajectory tracking problem. However, as pointed out earlier, mere kinematic control rarely suffices and often leads to degradation in control performance [3].

3.2 UT-Based Dual Adaptive Control

If the nonlinear dynamic functions $\boldsymbol{f}_k$ and $\boldsymbol{G}_k$ are assumed to be *perfectly* known, a non-adaptive computed-torque control law, like the one detailed in [10], solves the dynamic control problem (*i.e.* assuring that $\boldsymbol{\nu}_k$ tracks $\boldsymbol{\nu_c}_k$ reliably). However, it is an undeniable fact that in practice the robot dynamics; dependent on mass, inertia, friction and possibly several unmodelled phenomena; are typically unknown and may even change over time. In addition perfect sensor measurements are never available.

To address these complex issues of uncertainty, we propose a novel dual adaptive controller employing a MLP ANN trained online via an UKF algorithm in prediction mode. In contrast to the hitherto proposed innovation-based suboptimal dual adaptive laws [9,10], the control law we propose here employs the UT to approximate better the mean and covariance terms arising in the chosen cost function. Hence, the envisaged improvement is not solely due to the superior stochastic estimator employed to train the ANN (the UKF instead of the EKF), but also due to the dual adaptive law itself, as clarified further in the following treatment.

Neuro-Stochastic Estimator. To deal with the uncertainty and/or time-varying nature of the dynamic functions $\boldsymbol{f}_k$ and $\boldsymbol{G}_k$, we opt to assume that they are completely unknown to the controller and employ a stochastically trained ANN algorithm for their approximation in real-time.

A sigmoidal MLP ANN with one hidden layer, depicted in Figure 2, is used to approximate the nonlinear vector-valued function $\boldsymbol{f}_{k-1}$. Its output is given by

$$\tilde{\boldsymbol{f}}_{k-1} = \begin{bmatrix} \boldsymbol{\phi}^T(\boldsymbol{x}_{k-1}, \hat{\boldsymbol{a}}_k)\hat{\boldsymbol{w}}_{\boldsymbol{1}k} \\ \boldsymbol{\phi}^T(\boldsymbol{x}_{k-1}, \hat{\boldsymbol{a}}_k)\hat{\boldsymbol{w}}_{\boldsymbol{2}k} \end{bmatrix}, \tag{6}$$

in the light of the following statements:

Definition 1. $\boldsymbol{x}_{k-1} = [\boldsymbol{\nu}_{k-1} \quad 1]$ *denotes the ANN input. The augmented constant serves as a bias input.*

Definition 2. $\boldsymbol{\phi}(\cdot,\cdot)$ *is the vector of sigmoidal activation functions, whose i^{th} element is given by $\phi_i = 1/\left(1 + \exp\left(-\hat{\boldsymbol{s}}_{\boldsymbol{i}}^T \boldsymbol{x}\right)\right)$, where $\hat{\boldsymbol{s}}_{\boldsymbol{i}}$ is i^{th} vector element in the group vector $\hat{\boldsymbol{a}}$;* i.e. $\hat{\boldsymbol{a}} = \left[\hat{\boldsymbol{s}}_{\boldsymbol{1}}^T \quad \cdots \quad \hat{\boldsymbol{s}}_{\boldsymbol{L}}^T\right]^T$ *where L denotes the number of neurons. The time index has been dropped for clarity, and throughout the article the ˆ notation indicates that the operand is undergoing estimation.*

Definition 3. $\hat{\boldsymbol{w}}_{\boldsymbol{i}k}$ *represents the synaptic weight estimate vector of the connection between the neuron hidden layer and the i^{th} output element of the ANN.*

Assumption 3. *The input vector $\boldsymbol{x}_{k-1}$ is contained within a known, arbitrarily large compact set $\boldsymbol{\chi} \subset \mathbb{R}^2$.*

Moreover, it is known that $\boldsymbol{G}_{k-1}$ is a state-independent matrix with unknown elements (refer to (4)). Hence, its estimation does not require the use of an ANN. In addition it is a symmetric matrix, a property which is exploited to construct its estimate as follows

$$\tilde{\boldsymbol{G}}_{k-1} = \begin{bmatrix} \hat{g}_{1k-1} & \hat{g}_{2k-1} \\ \hat{g}_{2k-1} & \hat{g}_{1k-1} \end{bmatrix}, \tag{7}$$

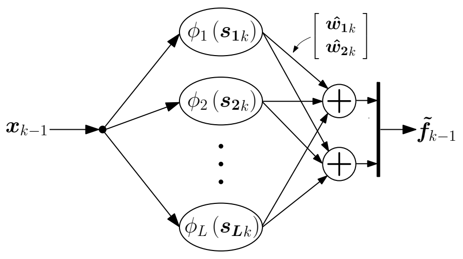


Fig. 2. Sigmoidal Multilayer Perceptron neural network

where $\hat{g}_{1_{k-1}}$ and $\hat{g}_{2_{k-1}}$ represent the estimates of the unknown elements in $\boldsymbol{G}_{k-1}$.

We opt to formulate the ANN weight-tuning task as a stochastic nonlinear estimation problem. The following preliminaries are required.

Definition 4. *The unknown parameters are grouped in a single vector* $\hat{\boldsymbol{z}}_k = \left[\hat{\boldsymbol{r}}_k^T \ \hat{\boldsymbol{g}}_k^T\right]^T$*, where* $\hat{\boldsymbol{r}}_k = \left[\hat{\boldsymbol{w}}_{\boldsymbol{1}_k}^T \ \hat{\boldsymbol{w}}_{\boldsymbol{2}_k}^T \ \hat{\boldsymbol{a}}_k^T\right]^T$ *and* $\hat{\boldsymbol{g}}_k = \left[\hat{g}_{1_{k-1}} \ \hat{g}_{2_{k-1}}\right]^T$.

Definition 5. *The measured output in the dynamic model (3) is denoted by* $\boldsymbol{y}_k = \boldsymbol{\nu}_k - \boldsymbol{\nu}_{k-1}$.

Assumption 4. *By the* Universal Approximation Theorem *of ANN, inside the compact set* $\boldsymbol{\chi}$*, the ANN approximation error is negligibly small when the estimate* $\hat{\boldsymbol{r}}_k$ *is equal to some unknown optimal vector denoted by* $\boldsymbol{r}_k^*$*. The* * *notation denotes optimality.*

In view of the stochastic adaptive approach taken in this work, the unknown optimal parameter vector $\boldsymbol{z}_k^*$ is treated as a random variable, with the initial condition $p(\boldsymbol{z}_0^*) \sim \mathcal{N}(\hat{\boldsymbol{z}}_0, \boldsymbol{P}_0)$, where the covariance $\boldsymbol{P}_0$ reflects the confidence in the initial guess $\hat{\boldsymbol{z}}_0$. Moreover, $\boldsymbol{z}_k^*$ is characterized as a stationary process corrupted by an artificial process noise $\boldsymbol{\rho}_k$, which aids convergence and tracking during estimation. In addition, observation uncertainty is catered for by augmenting a random measurement noise $\boldsymbol{\epsilon}_k$ to $\boldsymbol{y}_k$.

By (6), (7), all previous definitions and assumptions, it follows that the model in (3) can be represented in the following stochastic state-space form

$$\begin{aligned} \boldsymbol{z}_{k+1}^* &= \boldsymbol{z}_k^* + \boldsymbol{\rho}_k \\ \boldsymbol{y}_k &= \boldsymbol{h}\left(\boldsymbol{x}_{k-1}, \boldsymbol{\tau}_{k-1}, \boldsymbol{z}_k^*\right) + \boldsymbol{\epsilon}_k, \end{aligned} \tag{8}$$

where the vector-valued function $\boldsymbol{h}\left(\boldsymbol{x}_{k-1}, \boldsymbol{\tau}_{k-1}, \boldsymbol{z}_k^*\right)$ is nonlinear in the unknown optimal parameter vector $\boldsymbol{z}_k^*$, and is given by

$$\boldsymbol{h}\left(\boldsymbol{x}_{k-1}, \boldsymbol{\tau}_{k-1}, \boldsymbol{z}_k^*\right) = \tilde{\boldsymbol{f}}(\boldsymbol{x}_{k-1}, \boldsymbol{r}_k^*) + \tilde{\boldsymbol{G}}(\boldsymbol{g}_k^*)\boldsymbol{\tau}_{k-1}. \tag{9}$$

It is evident, from (9), that the use of the MLP ANN, which brings about certain practical advantages over other types of ANNs such as the Gaussian Radial Basis function (GaRBF) network as argued in [10], results in a *nonlinear* measurement equation in the stochastic state-space model (8) formulated for estimation. To address this issue, in a stochastic framework one has to employ a nonlinear recursive estimator. Conventionally, the EKF is used for this purpose [9,14,10]. However as shown in [11,12], the UKF can be a better alternative for stochastic nonlinear estimation. Its benefits over the EKF include, a derivative-free algorithm and superior accuracy in its approximations. For this reason, we opt to employ the UKF in predictive mode for the real-time estimation of $\boldsymbol{z}^*_{k+1}$ as follows.

Definition 6. *The* information state *denoted by I^k, consists of all measurements up to instant k and all the previous inputs.*

Assumption 5. *$\boldsymbol{\epsilon}_k$ and $\boldsymbol{\rho}_k$ are both zero-mean white Gaussian processes with covariances $\boldsymbol{R}_{\boldsymbol{\epsilon}}$ and $\boldsymbol{Q}_{\boldsymbol{\rho}}$ respectively. Moreover, $\boldsymbol{\epsilon}_k$, $\boldsymbol{\rho}_k$ and $\boldsymbol{z}^*_0$ are mutually independent $\forall k$.*

Lemma 1. *In the light of (8), Definition 6, and Assumption 5, it follows that $p(\boldsymbol{z}^*_{k+1}|I^k) \approx \mathcal{N}(\hat{\boldsymbol{z}}_{k+1}, \boldsymbol{P}_{k+1})$, where $\hat{\boldsymbol{z}}_{k+1}$ and $\boldsymbol{P}_{k+1}$ are computed at each control step according to Algorithm 1. Consequently, $\hat{\boldsymbol{z}}_{k+1}$ is considered to be the estimate of $\boldsymbol{z}^*_{k+1}$ conditioned on I^k, and $\boldsymbol{P}_{k+1}$ can be viewed as a measure of this estimate's uncertainty.*

Proof. The UKF algorithm in prediction mode, devised specifically for the purpose of this scheme, is effectively the standard UKF algorithm as presented in [12] for parameter estimation, with the difference that the measurement-update step precedes that for time-update. In addition, the time-update step is advanced by one sample to obtain $\hat{\boldsymbol{z}}_{k+1|k}$ at instant k. Hence, the proof of Lemma 1 follows directly that of the UKF (additive noise version) when applied to the nonlinear stochastic state-space model in (8). □

Lemma 2. *On the basis of Lemma 1, it follows that $p(\boldsymbol{y}_{k+1}|I^k)$ is approximately Gaussian with mean $\hat{\boldsymbol{y}}_{k+1}$ and covariance $\boldsymbol{P}_{\boldsymbol{yy}_{k+1}}$ given by:*

$$\hat{\boldsymbol{y}}_{k+1} = \hat{\boldsymbol{f}}_k + \hat{\boldsymbol{G}}_k \boldsymbol{\tau}_k, \tag{10}$$

$$\textit{where,}\ \ \hat{\boldsymbol{f}}_k = \sum_{i=0}^{2N} W_{mi} \mathbb{F}_{i,k+1|k}, \quad \hat{\boldsymbol{G}}_k = \tilde{\boldsymbol{G}}\left(\hat{\boldsymbol{g}}_{k+1}\right) \tag{11}$$

$$\textit{and}\quad \boldsymbol{P}_{\boldsymbol{yy}_{k+1}} = \sum_{i=0}^{2N} W_{ci}\left[\boldsymbol{D}_{\boldsymbol{f}_i} + \boldsymbol{D}_{\boldsymbol{G}_i}\boldsymbol{\tau}_k\right]\left[\boldsymbol{D}_{\boldsymbol{f}_i} + \boldsymbol{D}_{\boldsymbol{G}_i}\boldsymbol{\tau}_k\right]^T + \boldsymbol{R}_{\boldsymbol{\epsilon}} \tag{12}$$

$$\textit{where,}\ \ \boldsymbol{D}_{\boldsymbol{f}_i} = \mathbb{F}_{i,k+1|k} - \hat{\boldsymbol{f}}_k\ , \quad \boldsymbol{D}_{\boldsymbol{G}_i} = \mathbb{G}_{i,k+1|k} - \hat{\boldsymbol{G}}_k.$$

Proof. The equation of $\hat{\boldsymbol{f}}_k$ in (11) is derived by applying the UT to estimate the mean of $p\left(\tilde{\boldsymbol{f}}(\boldsymbol{x}_k, \boldsymbol{r}^*_{k+1})|I^k\right)$. The equation of $\hat{\boldsymbol{G}}_k$ in (11) is derived by employing the standard result in linear probability theory, namely $p(\boldsymbol{A}\boldsymbol{x}) = \boldsymbol{A}\bar{\boldsymbol{x}}$. This can be done since $\tilde{\boldsymbol{G}}_k$ is linear in the unknown parameters. To derive the equation of $\boldsymbol{P}_{\boldsymbol{yy}_{k+1}}$ in (12) one needs to advance (14) by one sampling instant, and substitute for $\mathbb{Y}_{i,k+1|k}$ and $\hat{\boldsymbol{y}}_{k+1}$, using the relations leading to (13) in the same algorithm. □

Algorithm 1. The UKF parameter-prediction algorithm

Given the previous prediction $\left(\hat{\boldsymbol{z}}_{k|k-1}, \boldsymbol{P}_{k|k-1}\right)$, denoted in short-form by $(\hat{\boldsymbol{z}}_k, \boldsymbol{P}_k)$, the following UKF algorithm (prediction mode) generates the new prediction $(\hat{\boldsymbol{z}}_{k+1}, \boldsymbol{P}_{k+1})$:

1) Sigma-points sampling and propagation:

$$\begin{aligned}
\mathcal{Z}_{k|k-1} &= \left[\hat{\boldsymbol{z}}_k \;\; \hat{\boldsymbol{z}}_k + \left(\gamma\sqrt{\boldsymbol{P}_k}\right) \;\; \hat{\boldsymbol{z}}_k - \left(\gamma\sqrt{\boldsymbol{P}_k}\right)\right] \\
\mathbb{F}_{k|k-1} &= \tilde{\boldsymbol{f}}(\boldsymbol{x}_{k-1}, \mathcal{R}_{k|k-1}),\; \mathbb{G}_{k|k-1} = \tilde{\boldsymbol{G}}(\mathcal{G}_{k|k-1}) \\
\mathbb{Y}_{k|k-1} &= \mathbb{F}_{k|k-1} + \mathbb{G}_{k|k-1}\boldsymbol{\tau}_{k-1} \\
\hat{\boldsymbol{y}}_k &= \sum_{i=0}^{2N} W_{m\,i}\mathbb{Y}_{i,k|k-1}
\end{aligned} \tag{13}$$

2) Measurement update and estimate prediction:

$$\boldsymbol{P}_{\boldsymbol{yy}\,k} = \sum_{i=0}^{2N} W_{c\,i}\left[\mathbb{Y}_{i,k|k-1} - \hat{\boldsymbol{y}}_k\right]\left[\mathbb{Y}_{i,k|k-1} - \hat{\boldsymbol{y}}_k\right]^T + \boldsymbol{R}_{\boldsymbol{\epsilon}} \tag{14}$$

$$\begin{aligned}
\boldsymbol{P}_{\boldsymbol{zy}\,k} &= \sum_{i=0}^{2N} W_{c\,i}\left[\mathcal{Z}_{i,k|k-1} - \hat{\boldsymbol{z}}_k\right]\left[\mathbb{Y}_{i,k|k-1} - \hat{\boldsymbol{y}}_k\right]^T \\
\boldsymbol{K}_k &= \boldsymbol{P}_{\boldsymbol{zy}\,k}\boldsymbol{P}_{\boldsymbol{yy}\,k}^{-1}, \quad \boldsymbol{i}_k = \boldsymbol{y}_k - \hat{\boldsymbol{y}}_k \\
\hat{\boldsymbol{z}}_{k+1} &= \hat{\boldsymbol{z}}_k + \boldsymbol{K}_k\boldsymbol{i}_k \\
\boldsymbol{P}_{k+1} &= \boldsymbol{P}_k - \boldsymbol{K}_k\boldsymbol{P}_{\boldsymbol{yy}\,k}\boldsymbol{K}_k^T + \boldsymbol{Q}_{\boldsymbol{\rho}}
\end{aligned}$$

where: $\mathcal{Z}^T = \left[\mathcal{R}^T \; \mathcal{G}^T\right]^T$, $\gamma = \sqrt{N+\lambda}$, N is the length of $\hat{\boldsymbol{z}}_k$, the scaling parameter $\lambda = \alpha^2(N+\kappa) - N$, constant α determines the spread of the sigma-points, constant κ is a secondary scaling parameter, the UT weights are given by: $W_{m0} = \frac{\lambda}{N+\lambda}$, $W_{c0} = W_{m0} + 1 - \alpha^2 + \beta$, and $W_{mi} = W_{ci} = \frac{1}{2(N+\lambda)}$ $(i = 1, \ldots, 2N)$, and β includes prior knowledge of the estimate's distribution. Moreover, in the UKF framework the linear algebra operation of adding a column vector to a matrix is defined as the addition of the vector to each column of the matrix. For further details, including guidelines for selecting the UKF scaling parameters, one is referred to [12].

UT-based Dual Adaptive Control Law. Algorithm 1 in the light of Lemma 1, constitutes the weight adaptation law for the novel UT-based MLP dual adaptive scheme. In addition, Lemma 2 provides a real-time update of the density $p(\boldsymbol{y}_{k+1}|I^k)$. This information is crucial in dual control since unlike HCE schemes, dual adaptive controllers do not ignore the uncertainty of the estimates, but employ it in the development of the control law itself, as follows.

The explicit-type suboptimal innovation-based performance index, adopted from [9], and modified to fit the multiple-input-multiple-output (MIMO) nonlinear scenario at hand is given by

$$\begin{aligned}
J_{inn} = E\Big\{ &\left(\boldsymbol{y}_{k+1} - \boldsymbol{y_d}_{k+1}\right)^T \boldsymbol{Q_1}\left(\boldsymbol{y}_{k+1} - \boldsymbol{y_d}_{k+1}\right) \\
&+ \left(\boldsymbol{\tau}_k^T\boldsymbol{Q_2}\boldsymbol{\tau}_k\right) + \left(\boldsymbol{i}_{k+1}^T\boldsymbol{Q_3}\boldsymbol{i}_{k+1}\right)\Big| I^k\Big\},
\end{aligned} \tag{15}$$

where $E\left\{\cdot|I^k\right\}$ is the mathematical expectation conditioned on I^k, and the following definitions apply:

Definition 7. *$\boldsymbol{y_d}_{k+1} = \boldsymbol{\nu_c}_{k+1} - \boldsymbol{\nu}_k$ is the reference vector of $\boldsymbol{y}_{k+1}$. To obtain $\boldsymbol{\nu_c}_{k+1}$ at instant k we advance the kinematic control law by one sampling interval as explained in [10].*

Definition 8. *Design parameter matrices $\boldsymbol{Q_1}$, $\boldsymbol{Q_2}$ and $\boldsymbol{Q_3}$ are diagonal and $\in \mathbb{R}^{2\times 2}$. Additionally: $\boldsymbol{Q_1}$ is positive definite, $\boldsymbol{Q_2}$ is positive semi-definite, and element-wise $-\boldsymbol{Q_1} \leq \boldsymbol{Q_3} \leq \boldsymbol{0}$.*

Remark 1. The design parameter $\boldsymbol{Q_1}$ is introduced to penalize tracking errors, $\boldsymbol{Q_2}$ induces a penalty on large control inputs, and $\boldsymbol{Q_3}$ affects the innovation vector so as to induce the dual adaptive feature characterizing this stochastic control law.

Theorem 1. *The control law minimizing performance index J_{inn} in (15), subject to the WMR dynamic model in (3), all previous definitions, assumptions, lemmas and remarks, is given by*

$$\boldsymbol{\tau}_k = \left(\hat{\boldsymbol{G}}_k^T \boldsymbol{Q_1} \hat{\boldsymbol{G}}_k + \boldsymbol{Q_2} + \boldsymbol{N_{GG}}_{k+1}\right)^{-1} \left(\hat{\boldsymbol{G}}_k^T \boldsymbol{Q_1} (\boldsymbol{y_d}_{k+1} - \hat{\boldsymbol{f}}_k) - \boldsymbol{n_{Gf}}_{k+1}\right), \quad (16)$$

where, $\boldsymbol{N_{GG}}_{k+1} = \sum_{i=0}^{2N} W_{ci} \boldsymbol{D_G}_i^T \boldsymbol{Q_4} \boldsymbol{D_G}_i$, $\boldsymbol{n_{Gf}}_{k+1} = \sum_{i=0}^{2N} W_{ci} \boldsymbol{D_G}_i^T \boldsymbol{Q_4} \boldsymbol{D_f}_i$ *and* $\boldsymbol{Q_4} = \boldsymbol{Q_1} + \boldsymbol{Q_3}$.

Proof. Given the approximate Gaussian distribution of $p(\boldsymbol{y}_{k+1}|I^k)$ specified in Lemma 2, and standard results from linear algebra involving matrices, it follows that within this scheme, (15) can be rewritten as

$$J_{inn} = \left(\hat{\boldsymbol{y}}_{k+1} - \boldsymbol{y_d}_{k+1}\right)^T \boldsymbol{Q_1} \left(\hat{\boldsymbol{y}}_{k+1} - \boldsymbol{y_d}_{k+1}\right) + \boldsymbol{\tau}_k^T \boldsymbol{Q_2} \boldsymbol{\tau}_k + \operatorname{tr}\left(\boldsymbol{Q_4} \boldsymbol{P_{yy}}_{k+1}\right).$$

By substituting for $\hat{\boldsymbol{y}}_{k+1}$ and $\boldsymbol{P_{yy}}_{k+1}$ in the above equation, using the relations in (10) and (12) respectively, it is possible to factorize J_{inn} completely in terms of $\boldsymbol{\tau}_k$. The resulting quadratic expression is differentiated with respect to $\boldsymbol{\tau}_k$ and then equated to zero in order to determine its stationary point. This leads to (16). The resulting Hessian matrix is given by $2\left(\hat{\boldsymbol{G}}_k^T \boldsymbol{Q_1} \hat{\boldsymbol{G}}_k + \boldsymbol{Q_2} + \boldsymbol{N_{GG}}_{k+1}\right)$, which by Definition 8 and the definition of $\boldsymbol{N_{GG}}_{k+1}$ in (16), is clearly positive definite. This means that the UT-based dual adaptive control law stated in Theorem 1 *minimizes* (15) uniquely, and the inverse term in (16) exists without exceptions. □

Remark 2. $\boldsymbol{Q_3}$ which appears in the control law (16) via $\boldsymbol{Q}_4$ acts as a weighting factor, where at one extreme, with $\boldsymbol{Q_3} = -\boldsymbol{Q_1}$, the controller completely ignores the estimates' uncertainty, resulting in HCE control, and at the other extreme, with $\boldsymbol{Q_3} = \boldsymbol{0}$, it gives maximum attention to uncertainty, which leads to cautious control. For intermediate settings of $\boldsymbol{Q_3}$, the controller strikes a compromise and operates in dual adaptive mode. It is well known that HCE control leads to large tracking errors and excessive control actions when the estimates' uncertainty is relatively high. On the other hand, cautious control is notorious for sluggish response and control turn-off [9]. Consequently, dual control exhibits superior performance by striking a balance between the two extremes.

4 Simulation and Experimental Results

In contrast to the case of deterministic controllers, to prove strict convergence for a dual adaptive nonlinear control law, which is inherently stochastic, is still considered to be an open problem within the research community. Hence in practice, as argued in [15], the performance of dual adaptive controllers is typically verified by computer simulations and less frequently by real-life experiments. In this regard, this section reports a number of simulation results, including those from a Monte Carlo comparative analysis, as well as several experimental results obtained from vigourous real-life tests on a physical mobile robot.

4.1 Simulation Results

The WMR in Figure 1 was simulated using the continuous-time model given by (1) and (2). To render the simulations more realistic, a number of parameters in the dynamic model of the robot, namely: $d,\ m_c,\ I_c$ and $\bar{\boldsymbol{F}}(\boldsymbol{\nu})$; were programmed to vary randomly about a set of nominal values from one simulation trial to another. These variations adhere to the physics of realistic arbitrarily generated scenarios, comprising various load configurations and frictional conditions. The nominal values for these parameters are: $d = 10$cm, $m_c = 32$kg, $I_c = 0.84$kgm^2 and $\bar{\boldsymbol{F}}(\boldsymbol{\nu}) = \boldsymbol{F_c}\boldsymbol{\nu}$, where $\boldsymbol{F_c}$ is a diagonal matrix of viscous friction coefficients with nominal diagonal values $[0.3, 0.3]$. The other parameters in the WMR model were set to: $b = 22.95$cm, $r = 6.25$cm, $m_w = 1$kg, $I_w = 0.002$kgm^2 and $I_m = 0.005$kgm^2. The control sampling interval T was set to 50ms, and a zero-mean Gaussian velocity measurement noise with covariance $0.0001\boldsymbol{I}$, where $\boldsymbol{I}$ denotes the identity matrix, was included.

Each simulation trial consists of eight consecutive controller simulations. The first six of these correspond to the three modes of operation; *i.e.* HCE ($\boldsymbol{Q_3} = -\boldsymbol{Q_1}$), cautious ($\boldsymbol{Q_3} = \boldsymbol{0}$) and dual ($\boldsymbol{Q_3} = -0.8\boldsymbol{Q_1}$); for each of the two adaptive schemes being compared, *i.e.* the UT-based and the EKF-based dual adaptive controllers proposed in this article and in [10] respectively. The remaining two trials correspond to: (1) a nominally tuned nonadaptive (NT-NA) dynamic controller, which is effectively a nonadaptive computed-torque controller, as presented in [10], that uses the nominal values of the varying dynamic parameters as if they were the true ones. One should note that this is the best a nonadaptive controller can do when the exact robot parameters are uncertain (very realistic); (2) a nonadaptive computed-torque controller which is perfectly tuned (PT-NA) to the exact values of the varying dynamic parameters. The latter is not realistic, and is used solely for the purpose of relative comparisons. In contrast, the HCE, cautious and dual adaptive controllers assume *no preliminary information* about the robot whatsoever, since closed-loop control is activated immediately with the initial parameter estimate vector $\hat{\boldsymbol{z}}_0$ selected at random from a zero-mean, Gaussian distribution with variance 0.0025.

For the sake of fair comparison *the same*: measurement noise sequence, reference trajectory, initial conditions, initial filter covariance matrix $\boldsymbol{P}_0 = 0.1\boldsymbol{I}$, artificial process noise covariance $\boldsymbol{Q_\rho} = 10^{-6}\boldsymbol{I}$, tracking error penalty $\boldsymbol{Q_1} = \boldsymbol{I}$, and control input penalty $\boldsymbol{Q_2} = \boldsymbol{0}$; are used for each of the eight controllers. The measurement noise sequence is randomly generated afresh for each trial. In addition, the sigmoidal MLP

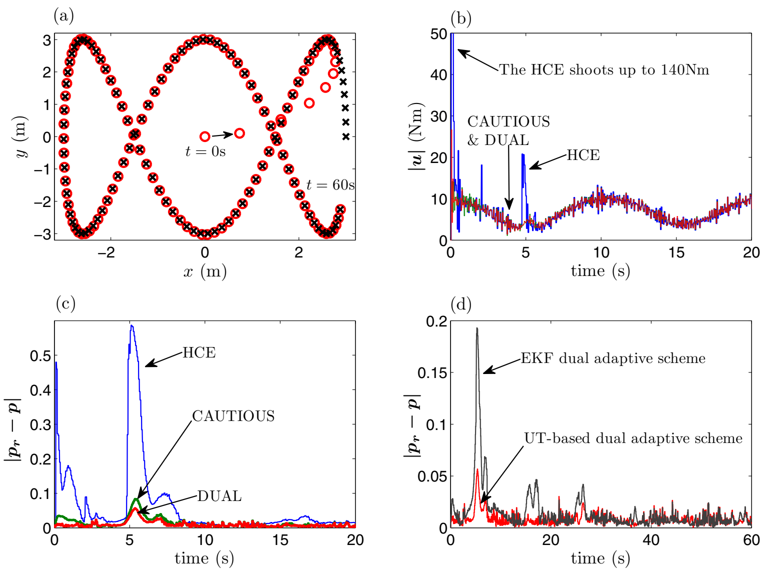
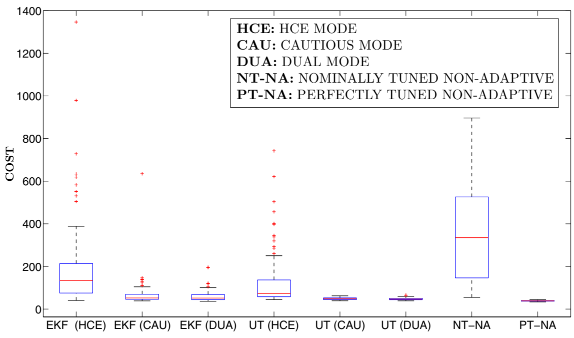


Fig. 3. (a): reference $(\times)$ and actual $(\bigcirc)$ trajectories (UT-based dual); (b): control input (UT-based 3 modes); (c): transient tracking error (UT-based 3 modes); (d): transient tracking error (UT-based dual vs EKF-based dual)

ANNs used in each of the two adaptive schemes are structured with five neurons in the hidden layer, *i.e.* $L = 5 \Rightarrow N = 27$. In the UT-based scheme, the UKF scaling parameters were set to: $\alpha = 1, \kappa = 0$ and $\beta = 2$.

Single Trial Results. A number of simulation results, typifying the performance of the three control modes of the proposed scheme are presented in Figure 3. Plot (a) depicts the WMR tracking a demanding reference trajectory, with a non-zero initial tracking error, controlled by the proposed UT-based adaptive controller in dual mode. It shows the excellent tracking performance of this controller, even when the trajectory reaches high speeds of around 1m/s. Plots (b) to (d) correspond to another simulation trial; purposely initiated with zero error conditions, so that any initial transient errors can be fully attributed to the convergence of the estimator. Plot (b) compares the Euclidian norm of the control input vector, for the three modes of the UT-based controller, during the first 20s. The very high transient control inputs of the HCE controller reflect the aggressive and incautious nature of this mode, which ignores completely the high uncertainty in the initial estimates. Plot (c) compares the Euclidian norm of the pose error vector, for the three modes of the UT-based controller, during the first 20s. This plot shows clearly how the HCE mode typically leads to high initial transient errors, while the dual mode exhibits the best transient performance. This is in accordance with Remark 2. Plot (d) compares the UT-based and the EKF-based controller (both in dual mode). This plot indicates that the former has better transient performance, while in steady-state the two controllers lead to the same negligible error.

Monte Carlo Comparative Results. A Monte Carlo simulation involving 100 simulation trials was performed. Each of the eight controller simulations in a trial corresponds to a trajectory time horizon of one minute in real time under the simulation conditions specified earlier, and with zero error initial conditions. After each controller simulation the following cost function C $= \sum_{k=0}^{k_{final}} |\boldsymbol{p_r}_k - \boldsymbol{p}_k|$ is computed. This serves as a measure of the tracking performance of each of the eight controllers operating under the same conditions, where lower values of C are naturally preferred. The eight cost distributions are depicted in the boxplot of Figure 4, and their respective mean and variance are tabulated and ranked in the same figure. These results indicate clearly that *in general* the UT-based dual adaptive controller brings about a significant improvement in tracking performance, not only over the non-adaptive controller which assumes nominal values for the robot dynamic parameters, but also over the EKF-based dual adaptive controller presented in [10]. Moreover, it is just as evident that within each of the two schemes, the dual control mode is even better than the cautious mode, as anticipated in Remark 2.



Controller	Mean	Var.	Rank
EKF-HCE	192	40225	6
EKF-CAU	67	3847	4
EKF-DUA	61	731	3
UT-HCE	140	32813	5
UT-CAU	48	30	2
UT-DUA	47	26	1
NT-NA	372	59614	7
PT-NA	39	5	-na-

Fig. 4. Monte Carlo simulation result: cost distributions (100runs)

It is also not surprising that the performance of the HCE modes is characterized by a high cost variance and several extreme outliers. This is the result of the complete lack of caution in the presence of high initial uncertainty, leading to high transient errors. An important observation is that each mode in the UT-based scheme is superior to the corresponding mode in the EKF-based scheme. We associate this to the better estimations of the UKF over the EKF in the ANN training algorithm, and to the better (second-order) approximations of the UT-based control law as opposed to the first-order approximation of the EKF-based control law.

4.2 Experimental Results

The UT-based dual adaptive controller proposed in this article was also implemented successfully on Neurobot [6], the WMR designed and built by the authors for the purpose of this research. A number of experimental results validating the proposed scheme

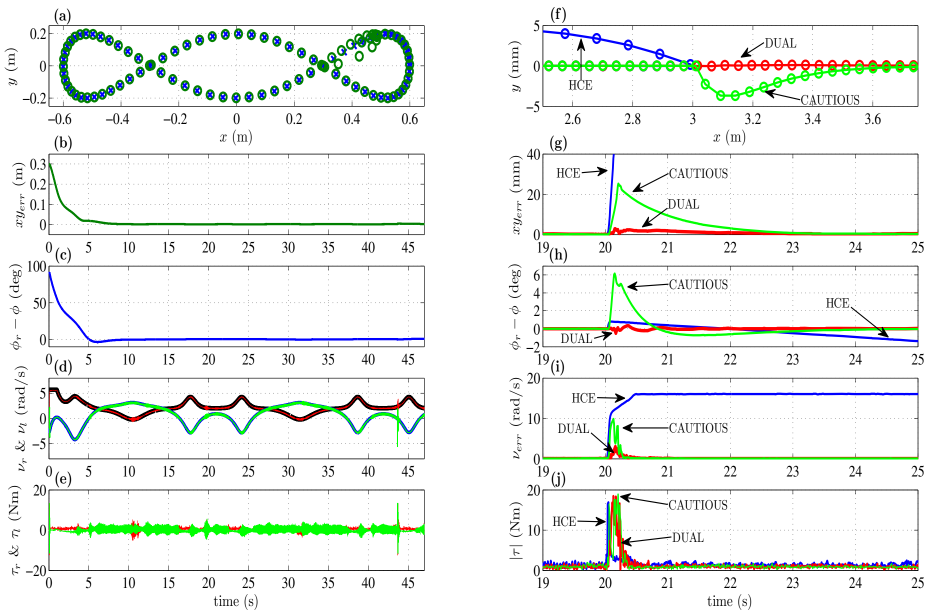


Fig. 5. (a): reference ($\times$) & actual ($\bigcirc$) trajectories; (b): position error; (c): orientation error; (d): wheel velocities & their references; (e): wheel torques; (f) to (j): HCE, cautious & dual comparative results

and confirming the simulation results in this section are provided in Figure 5. Plots (a) to (e) correspond to a challenging trajectory tracking experiment that tests the overall performance of the UT-based dual adaptive controller in a real-life application. From these results one should particularly note that: 1) Neurobot swiftly adapts to its own dynamics and simultaneously converges smoothly to the reference trajectory ((a) to (c)), 2) the actual and reference angular wheel velocities in (d) are practically superimposed and, 3) the wheel torques in (e) remain well bounded. Plots (f) to (j) correspond to a line trajectory test, with an artificially generated disturbance in the estimator which serves to compare the dual mode of the controller with its HCE and cautious counterparts. Specifically, at $t = 20$s, $\hat{z}_{k+1}$ is instantaneously set to some arbitrary value, hence erasing all the knowledge acquired up to this point in time by the ANN estimator. In this manner one can objectively compare the transient performance of the three control modes when faced with extremely high uncertainty in the robot dynamics. The results in Plots (f) to (j) clearly indicate that the dual controller exhibits the best transient performance, strongly confirming the arguments in Remark 2 and our simulation results.

5 Conclusions

The main contribution of the work reported in this article lies in the use of the UT to improve on the EKF-based dual adaptive dynamic WMR controller recently proposed in [10]. Specifically, the proposed UT-based dual adaptive scheme uses the UKF as a recursive ANN weight-tuning algorithm, and in addition features a novel dual adaptive law that employs the UT to better approximate the statistics of parameter distributions undergoing nonlinear transformations. The presented simulation and experimental results show clearly that the novel UT-based dual adaptive controller brings about significant improvements in performance, not only over the EKF-based dual adaptive scheme, but also over all other nondual and nonadaptive controllers tested in this work.

References

1. Kanayama, Y., Kimura, Y., Miyazaki, F., Noguchi, T.: A stable tracking control method for an autonomous mobile robot. In: Proc. IEEE Int. Conference of Robotics and Automation, Cincinnati, OH, pp. 384–389 (May 1990)
2. Canudas de Wit, C., Khennoul, H., Samson, C., Sordalen, O.J.: Nonlinear control design for mobile robots. In: Zheng, Y.F. (ed.) Recent Trends in Mobile Robots. Robotics and Automated Systems, pp. 121–156. World Scientific, Singapore (1993)
3. Fierro, R., Lewis, F.L.: Control of a nonholonomic mobile robot: Backstepping kinematics into dynamics. In: Proc. IEEE 34th Conference on Decision and Control (CDC 1995), New Orleans, LA, pp. 3805–3810 (December1995)
4. Corradini, M.L., Orlando, G.: Robust tracking control of mobile robots in the presence of uncertainties in the dynamic model. Journal of Robotic Systems 18(6), 317–323 (2001)
5. Wang, T.Y., Tsai, C.C.: Adaptive trajectory tracking control of a wheeled mobile robot via Lyapunov techniques. In: Proc. 30th Annual Conference of the IEEE Industrial Electronics Society, Busan, Korea, pp. 389–394 (November 2004)
6. Bugeja, M.K., Fabri, S.G.: Multilayer perceptron adaptive dynamic control of mobile robots: experimental validation. In: Bruyninckx, H., Kulich, L.P. (eds.) EuroSSC 2008, Springer Tracts in Advanced Robotics (STAR). pp. 165–174. Springer, Heidelberg (2008)
7. Åström, K.J., Wittenmark, B.: Adaptive Control, 2nd edn. Addison-Wesley, Reading (1995)
8. Fel'dbaum, A.A.: Optimal Control Systems. Academic Press, New York (1965)
9. Fabri, S.G., Kadirkamanathan, V.: Functional Adaptive Control: An Intelligent Systems Approach. Springer, London (2001)
10. Bugeja, M.K., Fabri, S.G.: Dual adaptive dynamic control of mobile robots using neural networks. IEEE Trans. Syst., Man, Cybern. B 39(1), 129–141 (2009)
11. Julier, S.J., Uhlmann, J.K.: A new extention of the Kalman filter to nonlinear systems. In: Proc. of AeroSense: The 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls (1997)
12. Wan, E.A., van der Merwe, R.: The unscented Kalman filter. In: Haykin, S. (ed.) Kalman Filtering and Neural Networks. Adaptive and Learning Systems for Signal Processing, Communications, and Control, pp. 221–280. John Wiley & Sons, Inc., Chichester (2001)
13. D'Amico, A., Ippoliti, G., Longhi, S.: A radial basis function networks apporach for the tracking problem of mobile robots. In: Proc. IEEE/ASME Int. Conference on Advanced Intelligent Mechatronics, Como, Italy, pp. 498–503 (2001)
14. Puskorius, G.V., Feldkamp, L.A.: Parameter-based Kalman filter training: Theory and implementation. In: Haykin, S. (ed.) Kalman Filtering and Neural Networks. Adaptive and Learning Systems for Signal Processing, Communications, and Control, pp. 23–67. John Wiley & Sons, Inc., Chichester (2001)
15. Filatov, N.M., Unbehauen, H.: Adaptive Dual Control: Theory and Applications. Springer, London (2004)

Multimodal System Based on Self-organizing Maps

Magnus Johnsson[1], Christian Balkenius[1], and Germund Hesslow[2]

[1] Lund University Cognitive Science, Kungshuset, Lundagård, 222 22 Lund, Sweden
{magnus.johnsson,christian.balkenius}@lucs.lu.se
[2] Department of Experimental Medical Science, Lund, BMC F10, 221 84 Lund, Sweden
germund.hesslow@med.lu.se
http://www.lucs.lu.se
http://www.mphy.lu.se/avd/nf/hesslow/index.html

Abstract. We present experiments with a multimodal system based on a novel variant of the Self-Organizing Map (SOM) called the Associative Self-Organizing Map (A-SOM). The A-SOM is similar to the SOM and develops a representation of its input space, but also learns to associate its activity with additional inputs, e.g. the activities of one or several external SOMs. This enables the modelling of expectations in one sensory modality due to the activity elicited in another modality. The paper presents the A-SOM algorithm generalized to an arbitrary number of (possibly delayed) associated activities together with simulation results with a multimodal sensory system and its extension to a system that also includes an action network. The simulation results were very encouraging and confirmed: The ability of the A-SOM to learn to associate the representations of its input space with the representations of the input spaces developed in two connected SOMs; The ability of the extended system to elicit proper activity in the action network; The simulations demonstrated good generalization ability.

Keywords: Self-Organizing Map, Neural network, Associative Self-Organizing Map, A-SOM, SOM, ANN, Expectations, Simulation hypothesis, Cognitive modelling, Cross-modal activation.

1 Introduction

A dramatic illustration of the interaction of different modalities can be seen in the McGurk-MacDonald effect. If you hear a person making the sound /ba/ but the sound is superimposed on a video recording on which you do not see the lips closing, you may hear the sound /da/ instead [11]. The neural mechanisms underlying such interaction between different sensory modalities are not known but recent evidence suggests that different primary sensory cortical areas can influence each other. Another familiar example is that the sensory information gained when the texture of an object is felt in the pocket can invoke visual images/expectations of the object.

An efficient multimodal perceptual system should be able to associate different modalities with each other in this way. This provides an ability to activate the subsystem for a modality even when its sensory input is limited or nonexistent as long as there are activities in subsystems for other modalities, which the subsystem has learned to associate with certain patterns of activity, which usually comes together with the patterns of activity in the other subsystems.

This paper explores a novel variant of the Self-Organizing Map (SOM) [9] called the Associative Self-Organizing Map (A-SOM). The A-SOM differs from earlier attempts to build associate maps such as the Adaptive Resonance Associative Map [13] and Fusion ART [12] in that all layers (or individual networks) share the same structure and uses topologically arranged representations. Unlike ARTMAP, the A-SOM also allows associations to be formed in both directions [4]. The A-SOM is an extension to the SOM, which learns to associate its activity with the activities of other SOMs. Previously versions of the A-SOM has been restricted to association with only one SOM [6]. This work was done in the context of haptic perception where a bio-inspired self-organizing texture and hardness perception system automatically learned to associate the representations of two submodalities (A-SOMs) with each other. The system employed a microphone based texture sensor and a hardness sensor that measured the compression of the explored material while applying a constant pressure. It successfully found associated representations of the texture and hardness submodalities when trained and tested with multiple samples gained from the exploration of a set of 4 soft and 4 hard objects of different materials with varying surface textures. However the version of the A-SOM used in this context was only able to associate with one SOM and its generalization ability was not explored at all.

The A-SOM explored in this paper has been generalized to enable association with an arbitrary number of SOMs. We have used the A-SOM in a multimodal system consisting of an A-SOM together with two ordninary SOMs. We tested the system with training and test sets constructed by selecting uniformly distributed random points from a subset of the plane, while employing Voronoi tessellations of this plane as well as of the grid of neurons constituting the A-SOM to determine its performance. We have also tested to add an action network (adapted by the delta rule) to the system, thus obtaining an architecture able to simulate both perceptions and actions. This implies an architecture which can elicit reasonable activity both in its perceptual networks (the A-SOM and the two SOMs) and in its action network. The implementation was done in C++ using the neural modelling framework Ikaros [1].

2 Associative Self-organizing Map

The Associative Self-Organizing Map (A-SOM), Fig. 1, can be considered as a SOM which learns to associate its activity with (possibly delayed) additional inputs, e.g. the activities of a number of external SOMs. It consists of an $I \times J$ grid of neurons with a fixed number of neurons and a fixed topology. Each neuron n_{ij} is associated with $r + 1$ weight vectors $w_{ij}^a \in R^n$ and $w_{ij}^1 \in R^{m_1}, w_{ij}^2 \in R^{m_2}, \ldots, w_{ij}^r \in R^{m_r}$. All the elements of all the weight vectors are initialized by real numbers randomly selected from a uniform distribution between 0 and 1, after which all the weight vectors are normalized, i.e. turned into unit vectors.

At time t each neuron n_{ij} receives $r + 1$ input vectors $x^a(t) \in R^n$ and $x^1(t - d_1) \in R^{m_1}, x^2(t - d_2) \in R^{m_2}, \ldots, x^r(t - d_r) \in R^{m_r}$ where d_p is the time delay for input vector $x^p, p = 1, 2, \ldots, r$.

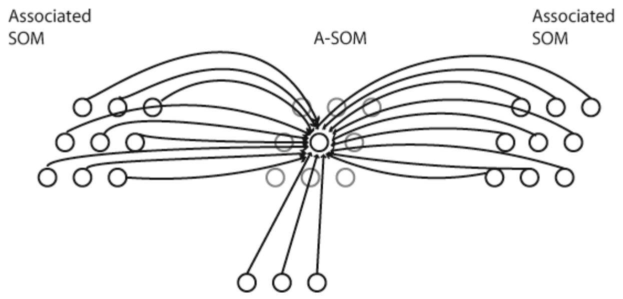


Fig. 1. The connectivity of the A-SOM neural network. During training each neuron in an A-SOM receives two kinds of input. One kind of input is the main (bottom-up) input, which corresponds to the input an ordinary SOM receives. The other kind of input is the activity of each neuron in one or more associated ancillary SOMs. In the fully trained A-SOM, activity can be triggered by either main input or by activity in one or several of the ancillary SOMs, or both.

The main net input s_{ij} is calculated using the standard cosine metric

$$s_{ij}(t) = \frac{x^a(t) \cdot w_{ij}^a(t)}{||x^a(t)||||w_{ij}^a(t)||}, \tag{1}$$

The activity in the neuron n_{ij} is given by

$$y_{ij}(t) = \left[y_{ij}^a(t) + y_{ij}^1(t) + y_{ij}^2(t) + \ldots + y_{ij}^r(t)\right] / (r+1) \tag{2}$$

where the main activity y_{ij}^a is calculated by using the softmax function [3]

$$y_{ij}^a(t) = \frac{(s_{ij}(t))^m}{\max_{uv}(s_{uv}(t))^m} \tag{3}$$

where u and v ranges over the rows and the columns of the neural network and m is the softmax exponent.

The ancillary activity $y_{ij}^p(t), \quad p = 1, 2, \ldots, r$ is calculated by again using the standard cosine metric

$$y_{ij}^p(t) = \frac{x^p(t-d_p) \cdot w_{ij}^p(t)}{||x^p(t-d_p)||||w_{ij}^p(t)||}. \tag{4}$$

The neuron c associated with the weight vector $w_c^a(t)$ most similar to the input vector $x^a(t)$, i.e. the neuron with the strongest main activation, is selected:

$$c = \arg\max_c\{|x^a(t) \cdot w_c^a(t)|\} \tag{5}$$

The weights w_{ijk}^a are adapted by

$$w_{ijk}^a(t+1) = w_{ijk}^a(t) + \alpha(t)G_{ijc}(t)\left[x_k^a(t) - w_{ijk}^a(t)\right] \tag{6}$$

where $0 \leq \alpha(t) \leq 1$ is the adaptation strength with $\alpha(t) \to 0$ when $t \to \infty$. The neighbourhood function $G_{ijc}(t) = e^{-\frac{||r_c - r_{ij}||}{2\sigma^2(t)}}$, where $r_c \in R^2$ and $r_{ij} \in R^2$ are location vectors of neurons c and n_{ij}, is a Gaussian function decreasing with time.

The weights $w_{ijl}^p, p = 1, 2, \ldots, r$, are adapted by

$$w_{ijl}^p(t+1) = w_{ijl}^p(t) + \beta x_l^p(t - d_p) \left[y_{ij}^a(t) - y_{ij}^p(t) \right] \tag{7}$$

where β is the constant adaptation strength.

All weights $w_{ijk}^a(t)$ and $w_{ijl}^p(t)$ are normalized after each adaptation.

3 Experiments and Results

3.1 Associating the A-SOM with Two Ancillary SOMs

We have evaluated the A-SOM by setting up a system consisting of one A-SOM and two connected SOMs (Fig. 2). To this end a set containing 10 training samples were constructed. This was done by randomly generating 10 points with a uniform distribution from a subset s of the plane $s = \{(x, y) \in R^2; 0 \leq x \leq 1, 0 \leq y \leq 1\}$ (Fig. 3, left). The selected points were then mapped to a subset of R^3 by adding a third constant element of 0.5, yielding a training set of three-dimensional vectors. The reason for this was that a Voronoi tessellation of the plane was calculated from the generated points to later aid in the determination of were new points in the plane were expected to invoke activity in the A-SOM. To make this Voronoi tessellation, which is based on a Euclidian metric, useful for this purpose with the A-SOM, which uses a metric based on dot product, the set of points in the plane has to be mapped so that the corresponding position vectors after normalization are unique. One way to accomplish such a mapping is by adding a constant element to each vector. The result of this is that each vector will have a unique angle in R^3. We chose the value 0.5 for the constant elements to maximize the variance of the angles in R^3.

The A-SOM was connected to two SOMs (using the same kind of activation as the main activation in the A-SOM, i.e. dot product with softmax activation) called SOM 1 and SOM 2, and thus also receive their respective activities as associative input, see Fig. 2. The A-SOM, SOM 1 and SOM 2 were then simultaneously fed with samples from the training set, during a training phase consisting of 20000 iterations. The two SOMs and the A-SOM could as well be fed by samples from three different sets, always receiving the same combinations of samples from the three sets (otherwise the system could not learn to associate them). This could be seen as a way of simulating simultaneous input from three different sensory modalities when an animal or a robot explores a particular object. Each of the three representations, the A-SOM and the two SOMs, consists of 15×15 neurons. The softmax exponent for each of them were set to 1000. Their learning rate $\alpha(0)$ was initialized to 0.1 with a learning rate decay of 0.9999 (i.e. multiplication of the learning rate with 0.9999 in each iteration), which means the minimum learning rate, set to 0.01, will be reached at the end of the 20000 training iterations. The neighbourhood radius, i.e. σ of the neighbourhood function $G_{ijc}(t)$ in eq. (6), was initialized to 15 for all three representations and shrunk to 1 during the

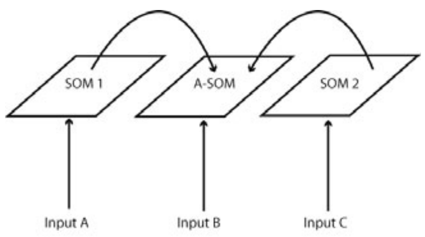


Fig. 2. Schematic depiction over the connections between the two SOMs and the A-SOM in the architecture of the test system used for this paper. The test system consist of three subsystems, which develop representations of sample sets from three input spaces (for simplicity we use the same input set for all three representations in the study for this paper). One of the representations (the A-SOM) also learns to associate its activity with the simultaneous activities of the two SOMs. This means proper activity can be invoked in the A-SOM of the fully trained system even if it does not receive any ordinary input. This is similar to cross-modal activation in humans, e.g. a tactile perception of an object can invoke an internal visual imagination of the same object.

20000 training iterations by using a neighbourhood decay of 0.9998 (i.e. multiplication of the neighbourhood radius with 0.9998 in each iteration). All three representations used plane topology when calculating the neighbourhood. The β for the associative weights in the A-SOM was set to 0.35.

After training the system was evaluated by feeding it with samples from the training set again to one, two or all three representations in all possible combinations. When a representation did not receive any input it was fed with null vectors instead (thus simulating the input of no signal from sensors of the modality of that representation). The centers of activity in the A-SOM as well as in the two SOMs were recorded for all these tests.

The result was evaluated by using the training set on the fully trained system. First we recorded the centers of activation in the A-SOM when fed by main input from the training set only (i.e. the two SOMs were fed with null vectors) and the centers of activation in the two SOMs. Then we calculated Voronoi tessellations for the centers of activation in all three representations (Fig. 4, uppermost row) to see if they could separate the samples and in particular if the A-SOM could separate the samples when fed by the activity of one or both of the SOMs only. If the center of activation for a particular sample in the training set were located in the correct Voronoi cell, this is considered as a successful recognition of the sample, because this means the center of activation is closer to the center of activation of the same object than to the center of activation of any other sample in the training set when the A-SOM is fed by main input only like an ordinary SOM. By comparing the Voronoi tessellations of the A-SOM and the two SOMs (Fig. 4) and the Voronoi tessellation of the plane for the training set (Fig. 3) we can see that the ordering of the Voronoi cells for the training set are to a large extent preserved for the Voronoi cells for the centers of activation in the A-SOM and the two SOMs. In Fig. 4 we can also see that all, i.e. 100% of the training samples are recognized in the A-SOM as long as at least one of the three representations received input.

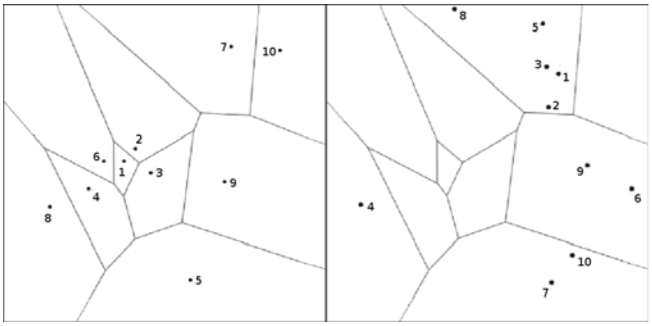


Fig. 3. Left: The Voronoi tessellation of the points used when constructing the training set used for the A-SOM and the two SOMs. This set was constructed by randomly generating 10 points from a subset of R^2 according to a uniform distribution. To make this Voronoi tessellation, which is based on a Euclidian metric, valid as a measure of proximity the training set had to be transformed by addition of a constant element to each sample vector. This is because the A-SOM using a dot product based metric and normalizing its input would consider all position vectors of a particular angle equal. By adding a constant element each point in the plane becomes a position vector in R^3 with a unique angle. Right: The same Voronoi tesselation but with the points used in the generalization test depicted. Also this set was mapped to a new set in R^3 by addition of a third constant element to each sample vector, and for the same reason as for the samples in the training set.

3.2 Generalization

To test if the system was able to generalize to a new set of samples, which it had not been trained with, we constructed a new set of 10 samples with the same method as for the training set. This generalization test set was used as input to the two SOMs and the A-SOM, i.e. each of these representations received the same sample simultaneously (or a null vector).

The generalization ability of the system was evaluated by feeding it with samples from the generalization set to one, two or all three representations in all possible combinations. When a representation did not receive any input it was fed with null vectors instead. The centers of activity in the A-SOM as well as in the two SOMs were recorded for all these tests.

The result was evaluated by now using the generalization set on the fully trained system. We recorded the centers of activation in the A-SOM when each of the SOMs were the only recipient of input, when both SOMs received input, when each of the SOMs and the A-SOM received input, when all three representations received input, and when only the A-SOM received input. As before a representation which did not receive input received null vectors (signifying the lack of sensory registration for that modality). We then looked at in which Voronoi cell the centre of activation was located in the A-SOM and in the SOMs for each sample of the generalization set. When a generalization sample belongs to the Voronoi cell for sample $k, k = 1, 2, \ldots, 10$ of the training set (see Fig. 3) and its activation in the A-SOM or one of the SOMs is located in the Voronoi cell for the centre of activation for the same training sample (see Fig. 4),

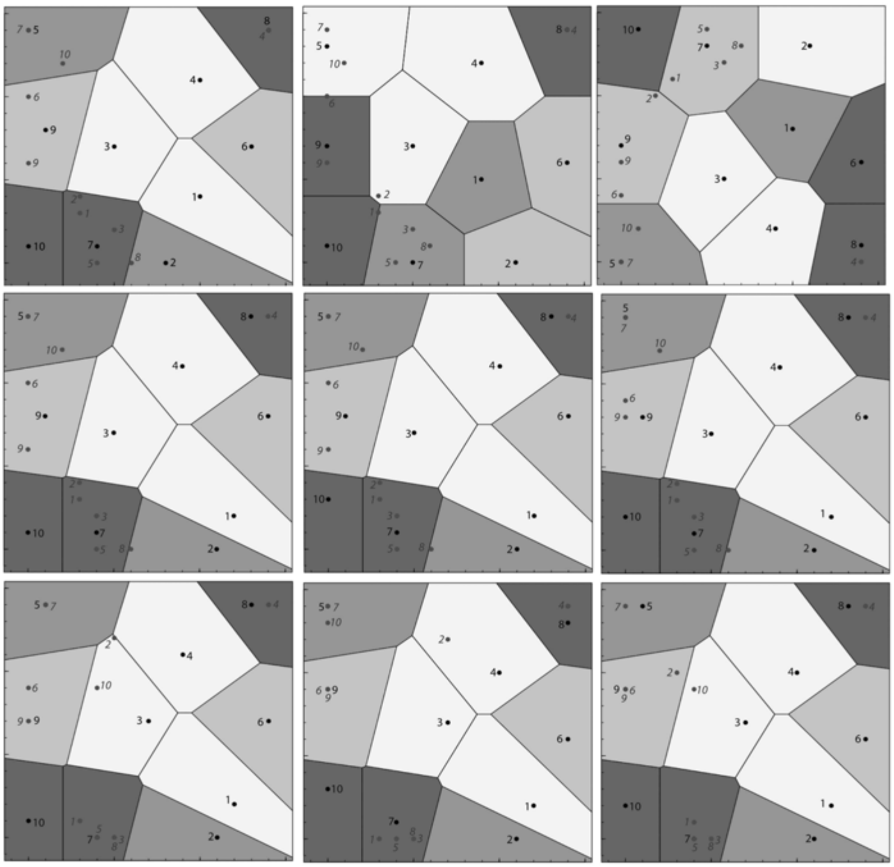


Fig. 4. The center of activation for different constellations of input to the fully trained system in the A-SOM and in the two SOMs. The centers of activation for the training samples and the generalization samples are denoted by numbers with normal and italic typefaces respectively. Upper row left: The A-SOM when only main input to the A-SOM is received. The Voronoi tessellation for these centers of activation has also been drawn. This is also true for the other images in this figure depicting activations in the A-SOM. Upper row middle: The SOM1 with the Voronoi tesselation for the training set drawn. Upper row right: The SOM2 with the Voronoi tesselation for the training set drawn. Middle row left: The A-SOM receiving main input and the activity of SOM1. Middle row middle: The A-SOM when receiving main input and the activity of SOM2. Middle row right: The A-SOM when receiving main input and the activities of SOM1 and SOM2. Lower row left: The A-SOM when receiving the activity of SOM1 only. Lower row middle: The A-SOM when receiving the activity of SOM2 only. Lower row right: The A-SOM receiving the activities of SOM1 and SOM2.

then we consider the centre of activation for the generalization sample to be properly located and we consider it to be successfully generalized.

Leftmost in the upper row of Fig. 4 we can see that the centers of activation for all the generalization samples besides sample 8 is within the correct Voronoi cell in the

A-SOM when it receives main input only. However that sample 8 is outside, and barely so, the correct Voronoi cell is probably not an indication that it is incorrect because the A-SOM consists of 225 neurons and is not a continuous surface but a discretized representation.

In the middle of the upper row of Fig. 4 we can see that all centers of activation for the generalization samples are correctly located in SOM1 besides 1 and 6 which are on the border to the correct Voronoi cell (but this should probably not be considered an indication of incorrectness for the same reason as mentioned above), and 2 which is located close to the correct Voronoi cell.

Rightmost of the upper row of Fig. 4 we can see that all centers of activation for the generalization samples are correctly located in SOM2 besides 2, which is located close to the correct Voronoi cell.

Leftmost in the middle row of Fig. 4 we can see that the centers of activation for all the generalization samples besides sample 8 (which should probably not be considered an indication of incorrectness for the same reason as mentioned above) is within the correct Voronoi cell in the A-SOM when it receives main input as well as the activity of SOM1 as input.

In the middle of the middle row of Fig. 4 we can see that the centers of activation for all the generalization samples besides sample 8 (which should probably not be considered an indication of incorrectness for the same reason as mentioned above) is within the correct Voronoi cell in the A-SOM when it receives main input as well as the activity of SOM2 as input.

Rightmost of the middle row of Fig. 4 we can see that the centers of activation for all the generalization samples besides sample 8 (which should probably not be considered an indication of incorrectness for the same reason as mentioned above) is within the correct Voronoi cell in the A-SOM when it receives main input as well as the activities of both SOM1 and SOM2 as input.

Leftmost of the lower row of Fig. 4 we can see that the centers of activation for all the generalization samples besides sample 2 and 10, i.e. 80%, is within the correct Voronoi cell in the A-SOM when it receives the activity of SOM1 as its only input.

In the middle of the lower row of Fig. 4 we can see that the centers of activation for all the generalization samples besides sample 2, i.e. 90%, is within the correct Voronoi cell in the A-SOM when it receives the activity of SOM2 as its only input.

Rightmost of the lower row of Fig. 4 we can see that the centers of activation for all the generalization samples besides sample 2 and 10, i.e. 80%, is within the correct Voronoi cell in the A-SOM when it receives the activities of SOM1 and SOM2 as its only input.

In Fig. 5 we can see a graphical representation of the activity in the two SOMs as well as total, main and ancillary activities of the A-SOM while receiving a sample from the generalization set. The lighter an area is in this depiction, the higher the activity is in that area.

3.3 Adding an Action Network

We have tested to add an action network to the system to get an architecture which can elicit reasonable activity both in its perceptual networks (the A-SOM and the two SOMs) and in its action network. This was done by fully connecting the A-SOM to

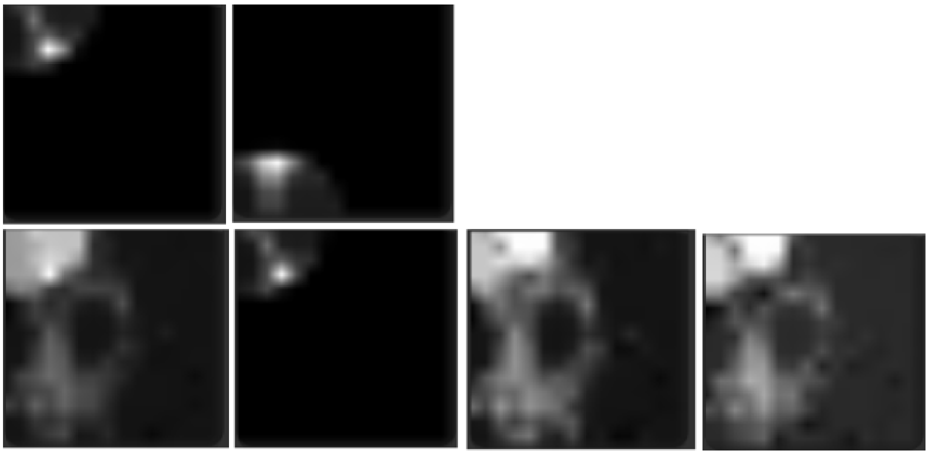


Fig. 5. Activations at a moment in the simulation. The lighter an area is in this depiction, the higher the activity is in that area. Upper row left: The activity in SOM1. Upper row right: The activity in SOM2. Lower row left: The total activity in the A-SOM. Lower row, the second image from the left: The main activity in the A-SOM. Lower row, the third image from the left: The ancillary activity in the A-SOM due to the activity in SOM1. Lower row right: The ancillary activity in the A-SOM due to the activity in SOM2.

an action network with feed-forward connections, i.e. the action network received the activity of the A-SOM as input. This extended architecture is depicted in Fig 6. It consists of a sensory part (the A-SOM and the two SOMs) and an action part (the action network).

The action network consists of an $I \times J$ grid of a fixed number of neurons that are adapted by the delta rule to get an activity that converges to the provided desired output.

Each neuron n_{ij} is associated with a weight vector $w_{ij} \in R^n$. All the elements of the weight vector are initialized by real numbers randomly selected from a uniform distribution between 0 and 1, after which the weight vector is normalized, i.e. turned into unit vectors.

At time t each neuron n_{ij} receives an input vector $x(t) \in R^n$.

The activity y_{ij} in the neuron n_{ij} is calculated using the standard cosine metric

$$y_{ij}(t) = \frac{x(t) \cdot w_{ij}(t)}{||x(t)||||w_{ij}(t)||}, \tag{8}$$

During the learning phase the weights w_{ijl}, are adapted by

$$w_{ijl}(t+1) = w_{ijl}(t) + \beta x_l(t) \left[y_{ij}(t) - d_{ij}(t)\right] \tag{9}$$

where β is the adaptation strength and $d_{ij}(t)$ is the desired activity for the neuron n_{ij}.

The sensory part of the extended system is equivalent to the original system and all settings of this part were equal to the settings of the original system. The learning rate β for the neurons in the action layer was set to 0.35.

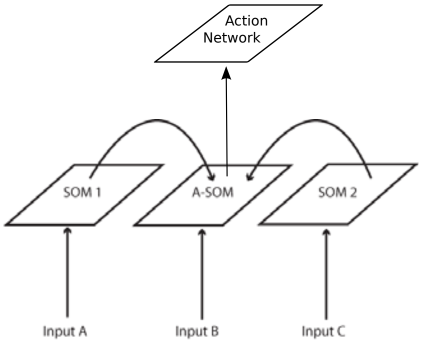


Fig. 6. Schematic depiction over the connections between the two SOMs, the A-SOM and the action network in the extended system

We have trained the extended architecture by using the same 10 training samples as before for both the SOMs and the A-SOM. Since the action network employs supervised learning (thus learning a desired activity associated with the input from the A-SOM) each training sample was associated with a desired activity provided to the action network during the training phase. The set of desired activities D consisted of 10-dimensional vectors, where one element in each was set to 1 and the other elements were set to 0, i.e. $D = \{(1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1)\}$.

The extended system was trained during 20000 iterations. After training the system was evaluated by (as in the case with the original system) feeding it with samples from the training set again to one, two or all three representations in the sensory part (the A-SOM and the SOMs) in all possible combinations. When a representation did not receive any input it was (as in the case with the original system) fed with null vectors instead (thus simulating the input of no signal from sensors of the modality of that representation).

The result was evaluated by using the training set on the fully trained system while recording whether the activity in the action network corresponded to the desired activity it was trained on or not. It turned out that in all 6 combinations of input (i.e. input to both SOMs and the A-SOM, one of the SOMs and the A-SOM, one SOM only, the A-SOM only) the activity elicited in the action network was 100% correct for all samples in the training set.

4 Discussion

We have presented and experimented with a novel variant of the Self-Organizing Map (SOM) called the Associative Self-Organizing Map (A-SOM), which develops a representation of its input space but also learns to associate its activity with an arbitrary number of additional inputs, e.g. the activities of other SOMs.

In our experiments we connected an A-SOM to two ancillary SOMs (thus getting a multimodal sensory system) and all these were trained and tested with a set of random samples of points from a subset of the plane. We also tested the generalization ability of the system by another set of random points generated from the same subset of the plane. In addition we also tested to add an action network to the system, thus obtaining an architecture able to simulate both perceptions and actions.

The ability of the A-SOM proved to be good, with 100% accuracy with the training set and about 80-90% accuracy in the generalization tests, depending on which constellation of inputs which was provided to the system. It was also observed that the generalization in the ordinary SOMs was not perfect. If this had been perfect the generalization ability would probably have been even better. This is probably a matter of optimizing the parameter settings.

The ability of the extended system to elicit proper activity in its action network proved to very good, with 100% correct in all 6 combinations of input.

In these experiments we connected an A-SOM with two SOMs, but we can see no reasons to why it should not be possible to connect an arbitrary number of A-SOMs to each other. Johnsson and Balkenius successfully connected two A-SOMs with each other in the context of a hardness/texture sensing system [6]. In the present study we used the same training set and the same generalization set as input for the A-SOM and for each of the two SOMs. This was for simplicity reasons and in particular because it made it easier to present the results and to relate the organizations of the SOMs and the A-SOM to each other.

It is interesting to speculate, and later test, whether there are any restrictions on the sets that are used as input to the different SOMs and A-SOMs in this kind of system. A reasonable guess would be that to learn to associate the activity arising from the training sets impose no restrictions on the training sets, but when it comes to generalization there would probably be one restriction. The restriction is that there should probably need to exist a topological function between the different input spaces so that the sequences of input samples from the different input spaces will invoke traces of activities over time in their respective SOM or A-SOM that in principle would be possible to map on each other by using only translations, rotations, stretching and twisting. Otherwise the generalization would be mixed up at least partially. The same would be true if the parameter setting implies the development of fragmentized representations.

Our system can be seen as a model of a neural system with two monomodal representations (the two SOMs) and one multimodal representation (the A-SOM) constituting a neural area that merges three sensory modalities into one representation, and (in the case of the extended system) one motor area.

The A-SOM actually develops several representations, namely one representation for its main input (the main activity) and one representation for each of the ancillary SOMs it is connected to (the ancillary activities), and one representation which merges these individual representations (the total activity). One could speculate whether something similar could be found in cortex, perhaps these different representations could correspond to different cortical layers.

Interaction between sensory modalities may be important for perceptual simulation. An idea that has been gaining popularity in cognitive science in recent years is that higher organisms are capable of simulating perception. In essence, this means that the perceptual processes normally elicited by some ancillary input can be mimicked by the brain [5]. There is now a large body of evidence supporting this contention. For instance, several neuroimaging experiments have demonstrated that activity in visual cortex when a subject imagines a visual stimulus resembles the activity elicited by a corresponding ancillary stimulus (for a review of this evidence see e.g. [10]; for a somewhat different interpretation, see [2].

A critical question here is how simulated perceptual activity might be elicited. One possibility is that signals arising in the frontal lobe in anticipation of consequences of incipient actions are sent back to sensory areas [5]. Another possibility is that perceptual activity in one sensory area can influence activity in another. The A-SOM provides a mechanism whereby sensory activity in an artificial system might be elicited or modulated by activity in a different sensory modality.

It should be noted that the model presented here is consistent with different views of how the sensory system is organized. The traditional view of sensory information processing has been that of a hierarchically organized system. Unimodal neurons in primary sensory cortex send signals to higher association areas where information from different modalities are eventually merged. The model presented in this paper is consistent with such a view. The A-SOM in fig. 2 could be seen as being a step higher in the sensory hierarchy than SOM-1 and SOM-2 and could project to other A-SOMs further up the hierarchy. However, recent neuroscientific evidence suggests that different primary sensory cortical areas can influence each other more directly. For instance, in a recent fMRI study [8] recently showed that visual stimuli can influence activity in primary auditory cortex. The associative SOM can serve as a model of such an organization as well. As an illustration, SOM-1 and A-SOM in fig. 2 could be located in an analog of a primary sensory cortical area, say an auditory area, and be influenced by signals from SOM-2, which could be located in a different, say visual, area.

The A-SOM can also be turned into a memory of perceptual sequences [7]. This can be done by connecting the total activity of the A-SOM back to itself as an ancillary input with a time delay. This works because then the ancillary weights will have learned to elicit activity based on the previous activity in the A-SOM. This could be useful for a robot to be able to continue even though its sensory input is interrupted. If that happens it can continue anyway by anticipating the sequence of perceptions likely to follow. We have done preliminary experiments with A-SOMs feeding their total activities back to themselves as time delayed ancillary input. In this way we have shown that a system of two A-SOMs (one with recurrent connections) was able to produce appropriate sequences of activity in both A-SOMs even when receiving no more input.

In the future we intend to test the A-SOM with several sets of recurrent ancillary input connections with different time delays, which might improve the capacity for remembering perceptual sequences. We will also try to extend and develop the presented ideas about the inclusion of action/motor neural networks in the system. In this way we hope to be able to explore the neuroscientific simulation hypothesis [5].

References

1. Balkenius, C., Morén, J., Johansson, B., Johnsson, M.: Ikaros: Building cognitive models for robots. In: Hülse, M., Hild, M. (eds.) Workshop on current software frameworks in cognitive robotics integrating different computational paradigms (in conjunction with IROS 2008), Nice, France, pp. 47–54 (2008)
2. Bartolomeo, P.: The relationship between visual perception and visual mental imagery: a reappraisal of the neuropsychological evidence. Cortex 38, 357–378 (2002)

3. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
4. Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., Rosen, D.B.: Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. IEEE Transactions on Neural Networks 3, 698–713 (1992)
5. Hesslow, G.: Conscious thought as simulation of behaviour and perception. Trends Cogn. Sci. 6, 242–247 (2002)
6. Johnsson, M., Balkenius, C.: Associating SOM representations of haptic submodalities. In: Ramamoorthy, S., Hayes, G.M. (eds.) Towards Autonomous Robotic Systems 2008, pp. 124–129 (2008)
7. Johnsson, M., Balkenius, C., Hesslow, G.: Neural network architecture for crossmodal activation and perceptual sequences. In: Papers from the AAAI Fall Symposium Biologically Inspired Cognitive Architectures 2009, pp. 85–86 (2009)
8. Kayser, C., Petkov, C.I., Augath, M., Logothetis, N.K.: Functional imaging reveals visual modification of specific fields in auditory cortex. J. Neurosci. 27, 1824–1835 (2007)
9. Kohonen, T.: Self-Organization and Associative Memory. Springer, Heidelberg (1988)
10. Kosslyn, S.M., Ganis, G., Thompson, W.L.: Neural foundations of imagery. Nature Rev. Neurosci. 2, 635–642 (2001)
11. McGurk, H., MacDonald, J.: Hearing lips and seeing voices. Nature 264, 746–748 (1976)
12. Nguyen, L.D., Woon, K.Y., Tan, A.H.: A self-organizing neural model for multimedia information fusion. In: International Conference on Information Fusion 2008, pp. 1738–1744 (2008)
13. Tan, A.H.: Adaptive resonance associative map. Neural Networks 8, 437–446 (1995)

Hydraulic Head Interpolation in an Aquifer Unit Using ANFIS and Ordinary Kriging

Bedri Kurtulus[1], Nicolas Flipo[2], Patrick Goblet[2], Guillaume Vilain[3], Julien Tournebize[4], and Gaëlle Tallec[4]

[1] Muğla University, Geological Engineering Department, 48000 Kotekli Muğla, Turkey
bkurtulus@mu.edu.tr
http://geoe.mu.edu.tr
[2] MINES ParisTech, Geosciences Department
35 rue Saint-Honoré,77305 Fontainebleau, France
Nicolas.Flipo@mines-paristech.fr
http://www.geosciences.mines-paristech.fr/
[3] CNRS/UPMC, UMR Sisyphe 7619, BP 105, Tour 55-56, 4 place Jussieu, 75252 Paris, France
[4] Cemagref, UR Hydrosystems and Bioprocesses, P.B. 44, 92163 Antony Cedex, France

Abstract. In this study, Ordinary Kriging (OK), and Adaptive Neuro Fuzzy based Inference System (ANFIS) are evaluated for assessing hydraulic head distribution in an aquifer unit covering 40 km^2. Cartesian coordinates of the samples were used as inputs of ANFIS. Calibrated models are used to interpolate the hydraulic head distribution on a 50 m square - grid. Both simulations have realistic pattern (R^2 > 0.97) even if OK performs slightly better than ANFIS at sampling location. The two methods capture different patterns. The Comparison of the two distributions allow for identifying area of estimate uncertainty, what can be used to improve the sampling network.

1 Introduction

A hydrosystem is defined as a ”part of space [where atmosphere overlap soil surface and subsurface] through which water flows. Physical and biogeochemical phenomena occur in all hydrosystem because of reactions due to water moving through a media” [1]. Many earth scientists (hydrologists, geologists, biogeochemists,) do interest in understanding the behaviour of such a complex system. Usually they first do experiments/observations in the field at specific locations and then try to distribute these observations/measurements in space and time using modelling techniques which are based on abstractions.

In this paper our focus is to distribute punctual hydraulic head measurements on a grid that covers a part of an experimental basin. One technique often used in earth sciences and especially in hydrogeology is kriging [2,3,4,5,6,7,8,9,10,11,12]. For a few years hydrologists started to use fuzzy logic and Adaptive neuro-fuzzy inference system (ANFIS) to estimate groundwater parameters [13], to predict reservoir level [14,15], river discharge [16,17,18,19,20,21,22,23] and karstic spring discharge [24], for groundwater management [25,26,27], and for assessing pollutant fluxes at the basin scale [28,29]. ANFIS was also used to model the hydrological cycle [30]. Nevertheless only

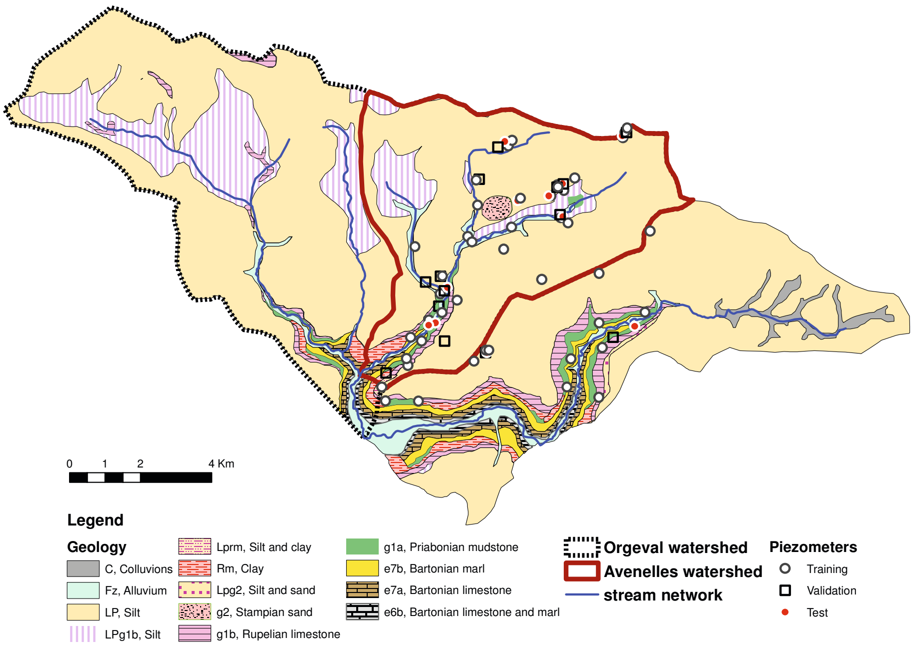


Fig. 1. Geological Map of the Orgeval watershed, location of wells and springs divided into training, validation and testing sets

few studies report the use of ANFIS to interpolate hydraulic head distribution in aquifers [31,7,32]. The goal of this work is to compare ordinary kriging (OK) and ANFIS in their ability to assess a hydraulic head distribution in a complex aquifer system.

2 Experimental Site

With an area of 104 km^2, the Orgeval experimental basin (Fig. 1) is located 70 km east from Paris [33,34]. Agriculture takes place on 80 % of its surface while the remaining 20 % are forested. The average annual air temperature is 9.7 ^{o}C. The annual mean rainfall is 706 mm, and the annual mean potential evaporation is 592 mm. The hydrological behaviour of the Orgeval basin is influenced by the aquifer system, which is composed of two main geological formations: the Oligocene (see Rupelian limestone, Fig. 1) and the Eocene (from Priabonian to Ypresian claystones, Fig. 1). These two aquifer units are separated by a clayey aquitard. Most of the basin is covered with table-land loess about 2-5m in thickness. These unconsolidated deposits are essentially composed of sand and loam lenses of low permeability but they seem to be more or less connected to the Rupelian limestone.

The basin is relatively flat with slopes increasing near the small valley at the river mouth (80 % of the territory spans between 130 and 190 m above mean sea level). In this work we will focus on hydraulic head distribution in the eastern part of the basin covering the Avenelles watershed (Fig. 1).

3 Data

The dataset is composed of three different types of data (Fig. 1). The first one consists of three piezometers which are permanently sampled. The second one consists of water levels measured in wells. The 41 wells were sampled on april 16, 2009 during a snapshot campaign. Our goal was to determine the hydraulic head distribution of the subsurface aquifer unit - silt connected to the rupelian limestone. Due to the complex geometry of the aquifer system at the outlet of the Avenelles basin and in the south-eastern part of the area of interest (Fig. 1), we needed to complete the piezometers and wells dataset in this part of the domain of interest. To do so we used a 100 $\times$ 100 m DEM of the top of the Priabonian mudstone. The elevation of the limit between Priabonian mudstone and rupelian limestone was then implemented inside the dataset as springs that are observed on the field. Finally the overall dataset is composed of 68 hydraulic heads.

4 Interpolation Methods

Ordinary Kriging. Geostatistics aims at providing quantitative descriptions of natural variables distributed in space and time [35]. Initially developed to address ore reserve evaluation issues in mining [36], it is now commonly applied to environmental sciences such as hydrogeology, air, water and soil pollution [37]. Geostatistics is used to characterize the spatial structure of the variable of interest by means of a consistent probabilistic model. This spatial structure is characterized by the variogram, which describes how the variability between sampled concentrations increases with the distance between the samples. A variogram model is fitted to the experimental variogram for subsequent analysis. The interpolation technique, known as kriging, provides the ”best”, unbiased, linear estimate of a regionalized variable at unsampled locations, where ”best” is defined in a least squares sense, as it aims to minimize the variance of estimation error [35]. As for the classical interpolations, the estimation by kriging of the concentration at any target cell is obtained by a linear combination of the available sample concentrations. The kriging differentiates only by the way of choosing the coefficients of this linear combination. Those coefficients are called kriging weights and depend on:

- the distances between the data and the target (like other classical interpolators),
- the distances between the original data themselves (data clustering),
- the spatial structure of the variable.

Exploratory data analysis, variogram fitting and kriging were performed using the Isatis software [38]. The basic tool used for kriging is the semi-variogram γ (eq. 1), defined as half the expectancy of deviation between values of samples separated by a distance h. In this case it quantifies the spatial variability of the variable $Z(x)$:

$$\gamma(h) = \frac{1}{2} E\left[(Z(x) - Z(x+h))^2 \right] \tag{1}$$

where $E[V]$ defines the mathematical average of the coordinates of the vector V. Let say, $Z^*(x)$ is the kriged value at location x, $Z(x_i)$ is the known value at location x_i, λ_i is the weight associated with the data, μ is the Lagrange multiplier, and $\gamma(x_i, x_j)$ is the value of variogram corresponding to a vector with origin in x_i and extremity in x_j. The general equation of kriging estimator is:

$$Z^*(x) = \sum_{i=1}^{n} \lambda_i Z(x_i) \tag{2}$$

In order to achieve unbiased estimations in kriging and to minimize the variance of estimates the following set of equations is solved simultaneously [39]:

$$\begin{cases} \sum_{i=1}^{N} \lambda_i = 1 \\ \sum_{j=1}^{N} \lambda_j \gamma(x_i, x_j) - \mu = \gamma(x_i, x) \qquad i = 1, \dots N \end{cases}$$

4.1 ANFIS

Theoretical Reminders. ANFIS [18,40,41,42,43,44,23] is a modelling technique which assumes that input and output data are ill-defined with uncertainty that can not be exactly assess with probability theory based on a two-valued logic. It uses fuzzy set theory, where a fuzzy set is a set of elements with an imprecise (vague) boundary [43,45]. A fuzzy set does not have a crisp boundary. That is, the transition from "belonging to the set" to "not belonging to the set" is gradual and is characterized by membership functions. A fuzzy set $A(x)$ is then represented by a pair of two things - the first one is the constituent elements x and their associated membership values $\mu_A(x)$ (that is their degree of belongingness):

$$A(x) = \{(x, \mu_A(x)), x \in X\} \tag{3}$$

Where X is the Universal set consisting of all possible elements. The membership function μ_A ranges between 0 and 1. If the value of the membership function is restricted to either 0 and 1, the fuzzy set is then reduced to classical crisp set with a known boundary. As stated by Jang [41], the fuzziness does not come from the randomness of the constituent members of the sets, but from the uncertain and imprecise nature of the abstract thoughts and concepts.

In ANFIS the relationship between input and output are expressed in the form of If-Then rules. ANFIS used for the present work is based on Sugeno fuzzy model [44] which formalizes a systematic approach to generate fuzzy rules from an input-output dataset. A typical fuzzy rule in a Sugeno fuzzy model has the format: If $x \in A$ and $y \in B$ then $z = f(x, y)$, where A and B are fuzzy sets in the antecedent and $f(x, y)$ is a crisp function in the consequent. Usually f is a polynomial function.

The architecture of the ANFIS is composed of five layers (Fig. 2). Each layer has a specific function. The first layer generates membership grades from linguistic labels translated in mathematical functions. It means that it defines the parameter of the

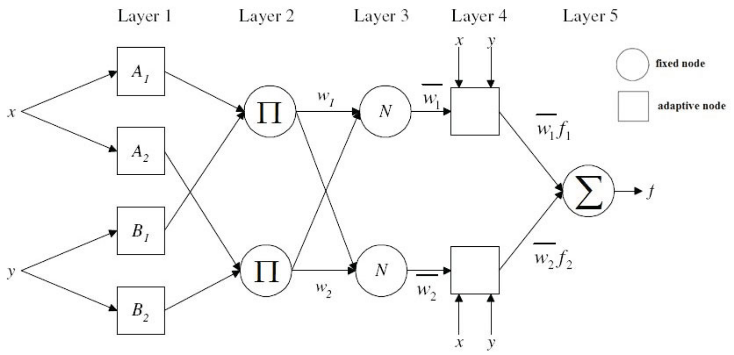


Fig. 2. ANFIS architecture for two inputs x, y. Layer 1: generates membership grades. Layer 2: Fuzzy rules. Layer 3: calculates weights of rules named firing strengths. Layer 4: product of the normalized firing strengths. Layer 5: fuzzy results transformed into a traditional output by summation.

membership functions. For instance, let consider a first-order Sugeno fuzzy inference system which contains two rules:

Rule 1: If $X \in A_1$ and $Y \in B_1$ then $f_1 = p_1 x + q_1 y + r_1$;
Rule 2: If $X \in A_2$ and $Y \in B_2$ then $f_2 = p_2 x + q_2 y + r_2$;

p_1, q_1,r_1, p_2, q_2,r_2 are defined in the first layer of the ANFIS (Fig. 2).

Each node i of layer 2 calculates the firing strength w_i of the i^{th} rule via multiplication $w_i = \mu_{A_i}(x)\mu_{B_i}(y)$.

Node i in the layer 3 calculates the ratio of the ith rule's firing strength divided by the total amount of all firing strengths: $\overline{w_i} = \frac{w_i}{\sum_j w_j}$.

Node i in the layer 4 calculates the contribution (weight) of the ith rule toward the overall output via multiplication:$\overline{W_i} = \overline{w_i} f_i$.

Finally layer 5 is made on a single node that computes the overall output as the summation of the contribution from each rule: $f(x, y) = \sum_i \overline{W_i} = \sum_i \overline{w_i} f_i$

ANFIS uses a hybrid learning algorithm that combines the back-propagation gradient descent and least squares methods to create a fuzzy inference system whose membership functions are iteratively adjusted according to a given set of input and output data [40]. For each iteration, the back propagation method involves minimization of an objective function using the steepest gradient descent approach in which the network weights and biases are adjusted by moving a small step in the direction of negative gradient. The iterations are repeated until a convergence criteria or a specified number of iterations is achieved. It has the advantage of allowing the extraction of fuzzy rules from numerical data and adaptively constructs a rule base.

Implementation of ANFIS. The neuro fuzzy model was developed using the ANFIS procedures of MATLAB [46]. In this study, a code is written in Matlab 7.0 for ANFIS using appropriate functions to calculate the best performance of the methods.

ANFIS is run on the same dataset as presented in section 3 Fig. 1. The dataset is divided in three subsets: a training one, a validation one and a test one.

Before using the model to interpolate unknown outputs (hydraulics head), its actual predictive performance must be tested by comparing outputs estimated by the calibrated models with known outputs. At each phase (training, validation), the ANFIS performance is measured by the determination of the coefficient of goodness-of-fit R^2, and the root mean square error (RMSE).

$$RMSE = \sqrt{E\left[(Z^*(x) - Z(x))^2\right]} \tag{4}$$

where E, Z^* and Z are previously defined (section 4).

Input data are XY coordinates of piezometers. Hydraulic head is the ANFIS output.

Input data are pre-processed to obtain input vectors for which coordinates are in the same range of variations. It is recommended to normalize the data between slightly offset values such as 0.1 and 0.9. In this work the preprocessing is done with a linear transformation. Let call X the input vector with n coordinates ranging from X$_{min}$ to X$_{max}$. Each coordinate (j) of the transformed variable Y is calculcated following the equation:

$$y_i = \frac{1}{X_{max} - X_{min}} \left(0.8X_i + 0.1X_{max} - 0.9X_{min}\right) \tag{5}$$

The selection of appropriate input parameters is a complex task. The first step is to determine the number of training and validation data. This selection is done iteratively to obtain the most similar training, validation and test sets [47] in terms of high and low values as well as concerning the statistical distribution:

- The area of interest is divided in four squares of equal size;
- If a square contains four points then two are selected for the training subset, one for the validation subset and one for the test subset. Else the square is divided in four squares of equal size and so on.

Finally the dataset was split into three subsets: 60 % of the data were assigned to the training set and the remaining to the validation and test set (20 % each). Early stopping criteria provided by the validation datasets are used to prevent overtraining. Generalized bell curves were used as membership functions (eq. 6):

$$f(x, a, b, c) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}} \tag{6}$$

5 Interpolation of Hydraulic Head: Results and Discussion

For each method (ANFIS and OK), the hydraulic head distribution was calculated on a 100 m square grid.

5.1 Ordinary Kriging

First of all the variographic clouds and the associated experimental variograms were calculated with different ranges (50 m, 100 m, 200 m and 1 km). They all reveal a clear linear structure (See Fig. 3 for a 250m range).

The fitted variogram is linear with a slope of 0.071 m (Fig. 3). The fitted variogram was then used to krige the hydraulic head at each center of the 100 m scare grid.

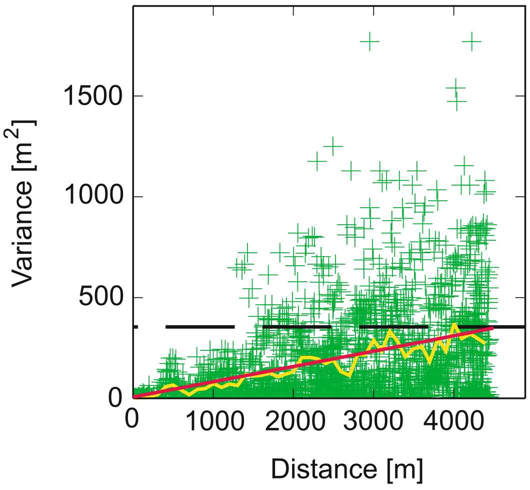


Fig. 3. Variogram cloud (green crosses), experimental variogram (yellow line) and modeled variogram (red line)

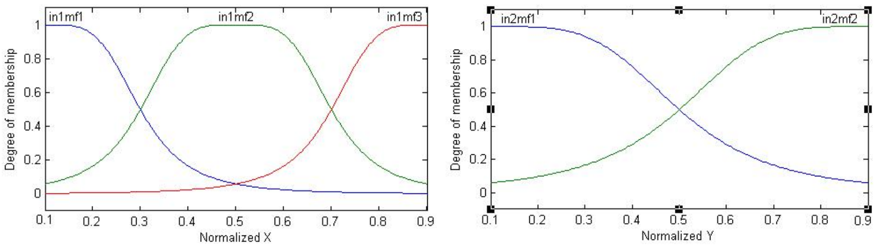


Fig. 4. Membership functions (after 44 iterations)

5.2 ANFIS

The best calibrated ANFIS model is obtained after 44 iterations. It contains 5 membership functions and 6 rules. Figure 4 shows the membership functions at the end of the learning phase.

5.3 Comparison of the Interpolation Methods

Table 1 shows statistics of both series (observed and predicted hydraulics head). The minimum, maximum, average and standard deviation values are of the same magnitude for simulations (whatever the techniques) and for the observed values. Even if the two methods match properly the data (Fig. 5) with R^2 of 0.97 for ANFIS and 0.99 for OK, the comparison of performances (Table 2) indicates a slight advantage for OK. Indeed RMSE for ANFIS and OK are 3.3 m and 0.8 m, respectively.

After being compared with observations at each sample location, each method is used to interpolate the dataset at each cell center of a 100×100 m grid (Fig. 5). The

Table 1. Observed and simulated data statistics. SD: standard deviation.

	Observed	ANFIS	OK
		Values at sampling points	
Mean [m]	139.49	139.47	139.33
Min [m]	102	107.73	102.42
Max [m]	179.85	181.03	179.47
SD [m]	20.05	19.91	19.90
		All Grid	
Mean [m]	-	101.78	102.42
Min [m]	-	193.65	181.05
Max [m]	-	143.83	141.89
SD [m]	-	20.54	18.14

Table 2. Statistics of errors for ANFIS and OK

	ANFIS	OK
RMSE[m]	139.49	139.47
ME [m]	102	107.73
MAE [m]	179.85	181.03
R^2 [m]	20.05	19.91

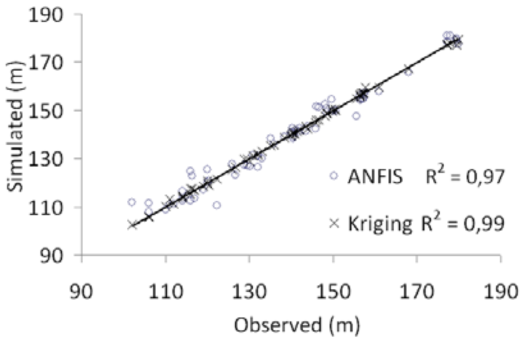


Fig. 5. Observed *vs.* simulated hydraulics heads

Average values of the whole set is 102.4 m for ANFIS whereas OK calculates an average of 101.8 m (Table 1). The standard deviation of the ANFIS interpolation increases (19.9 to 20.5 m) whereas the one of OK decreases (19.9 to 18.1 m).

Both simulations have realistic pattern except few details as local minima and maxima. ANFIS calculates a drainage pattern which looks less realistic than the OK one from a hydrogeological point of view. Even if OK performs slightly better than ANFIS, the latter seems to be a valuable way of interpolating hydraulic head distribution but not a more efficient method than OK as stated by Kholghi & Hosseini [31].

On the one hand, the fact that ANFIS under or overestimates few observed values in a larger proportion than OK (Fig. 5) may be due to the input variables (X and Y

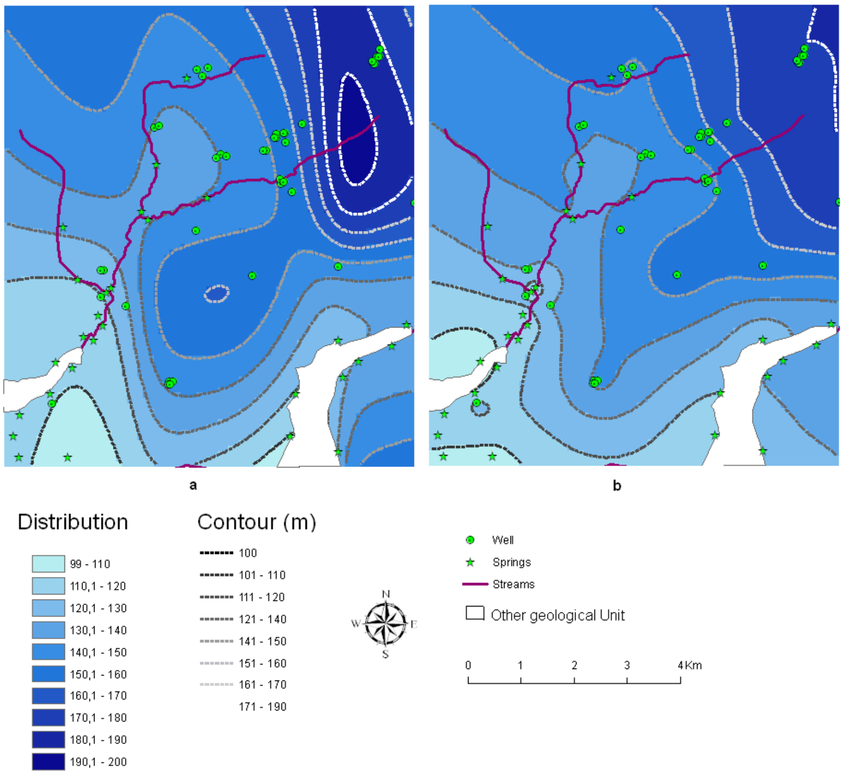


Fig. 6. (a) ANFIS interpolation and (b) OK interpolation

coordinates) of the ANFIS. Indeed these inputs do not have any physical meaning considering the hydraulic head distribution, which is partly driven by the river network. For further work one should test the euclidian distance to the river associated to only one coordinate (either X or Y) as input variables, or the use of a third input variable as the soil elevation for instance.

On the other hand, the less sampling points, the more sensitive is OK to the variogram that depends on the number of sampling points. In the Avenelles basin there are only 68 sampling points. The fitted variogram might entail uncertainty that leads to biased results [9]. Kriging might also be improved using a secondary variable as an external drift [5,48].

Nevertheless, the comparison of hydraulic head distributions calculated by OK and ANFIS (Fig. 6a & 6b) indicates that the two techniques capture the phenomenon in two different ways. At this point, one can use the estimation of the two hydraulic head distribution to improve measurement network based on discrepancies between each others (Fig. 7). The discrepancy map indicates in black and deep blue the area where sampling should be achieved in order to increase the dataset and then understand which method do perform best for the Orgeval aquifer unit.

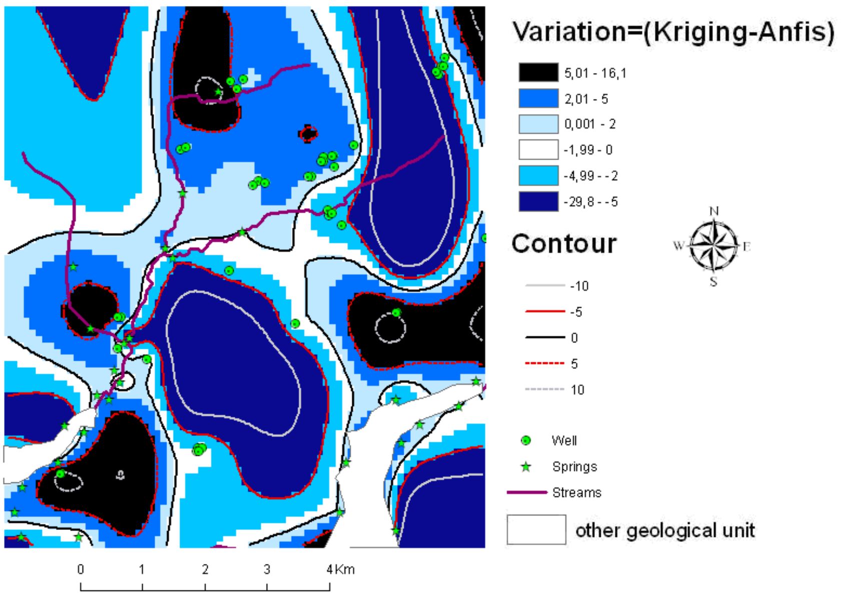


Fig. 7. Difference between hydraulic head distribution estimated with OK and ANFIS

6 Conclusions

The aim of this work was to interpolate hydraulic head distribution from punctual measurements at a basin scale in order to quantify groundwater resources based on a snapshot campaign. It was shown that OK slightly outperforms ANFIS and that the two methods can be improved. Even if both methods interpolate punctual hydraulic head measurements in two different patterns especially far from the sampling points where uncertainty is the highest, the difference between the two interpolated hydraulic head distribution can be used to identify new sampling locations and then improve the sampling network.

Acknowledgements. This work was funded by the ANR Carnot MINES "Neuro N'Eaudyssée", the Federation Ile-de-France for Research on the Environment (FIRE FR3020 CNRS & UPMC), the PIREN Seine research program, and TÜBITAK. It is also a contribution to the GIS Oracle that maintains the experimental basin of the Orgeval. We kindly thank the BRGM for providing the DEM of the top of the aquifer system.

References

1. Dacharry, M.: Encyclopedie. AXIS (1993)
2. Abedini, M., Nasseri, M., Ansari, A.: Cluster-based ordinary kriging of piezometric head in west texas/new mexico - testing of hypothesis. J. of Hydrology 351(3-4), 360–367 (2008)
3. Brochu, Y., Marcotte, D.: A simple approach to account for radial flow and boundary conditions when kriging hydraulic head fields for confined aquifers. Mathematical Geology 35(2), 111–139 (2003)
4. Buchanan, S., Triantafilis, J.: Mapping water table depth using geophysical and environmental variables. Ground Water 47(1), 80–96 (2009)

5. Desbarats, A.J., Logan, C.E., Hinton, M.J., Sharpe, D.R.: On the kriging of water table elevations using collateral information from a digital elevation model. Journal of Hydrology 255(1-4), 25–38 (2002)
6. Flipo, N., Jeannée, N., Poulin, M., Even, S., Ledoux, E.: Assessment of nitrate pollution in the Grand Morin aquifers (France): combined use of geostatistics and physically-based modeling. Environ. Pollut. 146(1), 241–256 (2007)
7. Kurtulus, B., Flipo, N., Vilain, G., Tournebize, J., Tallec, G., Goblet, P.: Comparison of ANFIS and ordinary kriging to assess hydraulic head distribution: the Orgeval case study. In: Proceedings of ICNC, Madeira, October 5-7 (2009)
8. Lyon, S., Seibert, J., Lembo, A., Walter, M., Steenhuis, T.: Geostatistical investigation into the temporal evolution of spatial structure in a shallow water table. Hydrol. Earth Syst. Sci. 10(1), 113–125 (2006)
9. Pardo-Igúzquiza, E., Chica-Olmo, M., Garcia-Soldado, M., Luque-Espinar, J.A.: Using semivariogram parameter uncertainty in hydrogeological applications. Ground Water 47(1), 25–34 (2009)
10. Renard, F., Jeannee, N.: Estimating transmissivity fields and their influence on flow and transport: The case of champagne mounts. Water Resources Research 44, 1–12 (2008)
11. Sun, Y., Kang, S., Li, F., Zhang, L.: Comparison of interpolation methods for depth to groundwater and its temporal and spatial variations in the minqin oasis of northwest china. Environmental Modelling & Software 24(10), 1163–1170 (2009)
12. Theodossiou, N., Latinopoulos, P.: Evaluation and optimisation of groundwater observation networks using the kriging methodology. Environmental Modelling & Software 21(7), 991–1000 (2006)
13. Ayvaz, M.T., Karahan, H., Aral, M.M.: Aquifer parameter and zone structure estimation using kernel-based fuzzy c-means clustering and genetic algorithm. J. of Hydrology 343, 240–253 (2007)
14. Chang, Y., Chang, L.C., Chang, F.J.: Intelligent control for modeling of real-time reservoir operation, part II: artificial neural network with operating rule curves. Hydrological Processes 19, 1431–1444 (2005)
15. Chang, F.J., Chang, Y.T.: Adaptive neuro-fuzzy inference system for prediction of water level in reservoir. Advances in Wat. Res. 29, 1–10 (2006)
16. Chidthong, Y., Tanaka, H., Supharatid, S.: Developing a hybrid multi-model for peak flood forecasting. Hydrological Processes 23, 1725–1738 (2009)
17. El-Shafie, A., Taha, M.R., Noureldin, A.: A neuro-fuzzy model for inflow forecasting of the Nile river at Aswan high dam. Water Resour. Manage. 21, 533–556 (2007)
18. Firat, M., Gungor, M.: River row estimation using adaptive neuro fuzzy inference system. Mathematics and Computers in Simulation 75, 87–96 (2007)
19. Firat, M., Gungor, M.: Hydrological time-series modelling using an adaptive neuro-fuzzy inference system. Hydrological Processes 22, 2122–2132 (2008)
20. Firat, M.: Comparison of artificial intelligence techniques for river flow forecasting. Hydrol. Earth Syst. Sci. 12, 123–138 (2008)
21. Hong, Y.S.T., White, P.A.: Hydrological modeling using a dynamic neuro-fuzzy system with on-line and local learning algorithm. Advances in Wat. Res. 32, 110–119 (2009)
22. Nayak, P., Sudheer, K., Ragan, D., Ramasastri, K.: A neuro-fuzzy computing technique for modeling hydrological time series. J. of Hydrology 291, 52–66 (2004)
23. Wang, W.C., Chau, K.W., Cheng, C.T., Qiu, L.: A comparison of performance of several artificial intelligence methods for forecasting monthly discharge time series. J. of Hydrology 374, 294–306 (2009)
24. Kurtulus, B., Razack, M.: Modeling daily discharge responses of a large karstic aquifer using soft computing methods: Artificial neural network and neuro-fuzzy. J. of Hydrology 381, 101–111 (2010)

25. Chu, H.J., Chang, L.C.: Application of optimal control and fuzzy theory for dynamic groundwater remediation design. Water Resour. Manage. 23, 647–660 (2009)
26. Firat, M., Turan, M.E., Yurdusev, M.A.: Comparative analysis of fuzzy inference systems for water consumption time series prediction. J. of Hydrology 374, 235–241 (2009)
27. Yurdusev, M.A., Firat, M.: Adaptive neuro fuzzy inference system approach for municipal water consumption modeling: An application to izmir, turkey. J. of Hydrology 365, 225–234 (2009)
28. Bárdossy, A., Haberlandt, U., Krysanova, V.: Automatic fuzzy-rule assessment and its application to the modelling of nitrogen leaching for large regions. Soft Computing 7, 370–385 (2003)
29. Marcé, R., Comerma, M., García, J., Armengo, J.: A neuro-fuzzy modeling tool to estimate fluvial nutrient loads in watersheds under time-varying human impact. Limnol. Oceanogr.: Methods 2, 342–355 (2004)
30. Bárdossy, A.: The use of fuzzy rules for the description of elements of the hydrological cycle. Ecol. Model. 85, 59–65 (1996)
31. Kholghi, M., Hosseini, S.M.: Comparison of groundwater level estimation using neuro-fuzzy and ordinary kriging. Environ. Model Assess. 14(6), 729–737 (2009)
32. Lin, G.F., Chen, L.H.: A spatial interpolation method based on radial basis function networks incorporating a semivariogram model. Journal of Hydrology 288(3-4), 288–298 (2004)
33. Anctil, F., Filion, M., Tournebize, J.: A neural network experiment on the simulation of daily nitrate-nitrogen and suspended sediment fluxes from a small agricultural catchment. Ecol. Model. 220, 879–887 (2009)
34. Flipo, N., Even, S., Poulin, M., Théry, S., Ledoux, E.: Modelling nitrate fluxes at the catchment scale using the integrated tool CAWAQS. Sci. Total Environ. 375, 69–79 (2007)
35. Chilés, J.P., Delfiner, P.: Geostatistics: modeling spatial uncertainty. Wiley, New York (1999)
36. Isaaks, E., Srivastava, R.: An introduction to applied geostatistics, p. 561. Oxford University Press, Oxford (1989)
37. Goovaerts, P.: Geostatistics for natural ressources evaluation, p. 181. Oxford University Press, Oxford (1997)
38. Geovariances: Isatis Software Manual, 5 edn., Geovariances and Ecole Nationale Supérieure des Mines de Paris, p. 710 (2004)
39. Chauvet, P.: Aide-mémoire de géostatistique linéaire. Ecole Nationale Supérieure des Mines de Paris (1999)
40. Jang, J.: ANFIS adaptive-network-based fuzzy inference systems. IEEE Trans. Systems, Man Cybern. 23(3), 665–685 (1993)
41. Jang, J.: Neuro-fuzzy modeling and control. Proceedings of the IEEE 833, 378–406 (1995)
42. Jang, J.: Input selection for ANFIS learning. In: IEEE International Conference on Fuzzy Systems, vol. 2, pp. 1493–1499 (1996)
43. Pratihar, D.: Soft Computing. Alpha Science International Ltd (2008)
44. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Systems Man and Cybernetics 15(1), 116–132 (1985)
45. Zadeh, L.: Fuzzy sets. Information and Control 8, 338–353 (1965)
46. Demuth, H., Beale, M.: Neural networks toolbox user guide. Technical report, Mathworks Inc. (2003)
47. Heuvelmans, G., Muys, B., Feyen, J.: Regionalisation of the parameters of a hydrological model: Comparison of linear regression models with artificial neural nets. Journal of Hydrology 319(1-4), 245–265 (2006)
48. Rivest, M., Marcotte, D., Pasquier, P.: Hydraulic head field estimation using kriging with an external drift: A way to consider conceptual model information. Journal of Hydrology 361(3-4), 349–361 (2008)

Predicting NN5 Time Series with Neurosolver

Andrzej Bieszczad

California State University Channel Islands, One University Drive
Camarillo, CA 93012, U.S.A.
andrzej_bieszczad@csuci.edu

Abstract. Neurosolver is a neuromorphic planner and a problem solving system. It was tested on several problem solving and planning tasks such as re-arranging blocks and controlling a software-simulated artificial rat running in a maze. In these tasks, the Neurosolver created models of the problem as temporal patterns in the problem space. These behavioral traces were then used to perform searches and generate actions. In this paper, we present an analysis of the capabilities of the Neurosolver to predict data points in time series. We report on testing those capabilities on the sample data sets that were made available for the neural network forecasting competition NN5 [14]. We conclude with a brief description of several ideas that we are currently applying to the data sets posted for the 2010 competition, NN GC1.

Keywords: Forecasting, Neural network, Neuromorphic systems, General problem solvers, Predicting future, Behavioral patterns.

1 Introduction

The goal of the research that led to the original introduction of Neurosolver, as reported in [1], was to design a neuromorphic device that would be able to solve problems in the framework of the state space paradigm [2]. In that paradigm, the states of a system are expressed as points in an n-dimensional space. Trajectories in such spaces formed by state transitions represent behavioral patterns of the system. A problem is presented in this paradigm as a pair of two states: the current state and the desired, or goal, state. A solution to the problem is a trajectory between the two states in the state space. Fundamentally, we are asking how to attain the goal state of the system given its starting state; in other words, we are searching for a sequence of state transitions leading from the start state to the goal state.

The Neurosolver can solve such problems by traversing the recorded trajectories as described in section 2. In this paper, we demonstrate how the very trajectories can be used for forecasting.

The original research on Neurosolver modelling was inspired by Burnod's monograph on the workings of the human brain [3]. The class of systems that employ state spaces to present and solve problems has its roots in the early stages of AI research that derived many ideas from the studies of human information processing; e.g., on General Problem Solver (GPS) [2]. This pioneering work led to very interesting problem solving (e.g. SOAR [4]) and planning systems (e.g. STRIPS [5].

The Neurosolver employs activity spreading techniques that have their root in early work on semantic networks (e.g., [6] and [7]).

The behavior of the Neurosolver bears similarities to stochastic processes [8]; especially discrete Markov models [9 10]. The Neurosolver learning algorithm resembles the forward-backward algorithm [11].

2 Neurosolver

2.1 Neurosolver as a GPS

The Neurosolver is a network of interconnected nodes. Each node is associated with a state in a problem space. In its original application, the Neurosolver is presented with a problem by two signals: the goal associated with the desired state and the sensory signal associated with the current state. A sequence of firing nodes that the Neurosolver generates represents a trajectory in the state space. Therefore, a solution to the given problem is a succession of firing nodes starting with the current node and ending with the goal node.

The node used in the Neurosolver is based on a biological cortical column (references to the relevant neurobiological literature can be found in [1]). It consists of two divisions: the upper and the lower, as illustrated in Figure 1. The upper division is a unit integrating internal signals from other upper divisions and from the control center providing the limbic input (i.e., a goal or - using more psychological terms - a drive or desire). The activity of the upper division is transmitted to the lower division where it is subsequently integrated with signals from other lower divisions and the thalamic input. The upper divisions constitute a network of units that propagate search activity from the goal, while the lower divisions constitute a network of threshold units that integrate search and sensory signals, and generate sequences of firing nodes. The output of the lower division is the output of the whole node. An inhibition mechanism prevents cycles and similar chaotic behavior. Simply, a node stays desensitized for a certain time after firing.

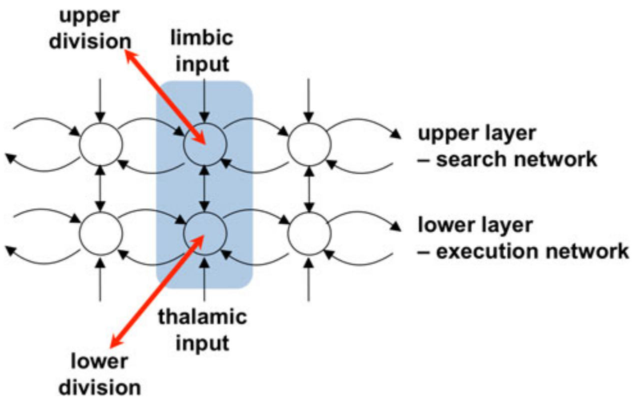


Fig. 1. An artificial cortical column

2.2 Neurosolver as a Forecaster

Normally, in the goal-oriented problem solving the flow of activity from the upper to the lower division is limited. This mode of operation can be described as exploration of possibilities and looking for environmental cues. The cues come as thalamic input from the sensory apparatus. Often though, we operate without far reaching goals forcing our brains to make predictions based on the knowledge of the past and the currently observed facts. In the Neurosolver, similar phenomenon may be observed if the activity in the upper division is gradually increased, and at the same time is allowed to be transmitted in its entirety from the upper to the lower division. Assuming that that activity is allowed to grow above the firing threshold level hosted by the lower division, a node may fire without extra signals from the sensors, or even in absence of the thalamic input whatsoever. In this paper, we explore this capability to predict future outcomes based on the statistical model created by the Neurosolver.

3 Data Sets

We presented some initial ideas on using the Neurosolver in the forecasting capacity at ISF'2008 [12]. We were encouraged to test the ideas on the data set that was used for the NNx competition. The last published data set available at the time of initiating experiments reported here was for the NN5 contest held in 2008, so that's what we used. The neural network forecasting competition is an ongoing event, so there are newer sets available currently.

The NN5 data set is a collection of records of daily withdrawals from a number of ATM machines in England over a two-year period. A set from an individual machine is divided into a larger training part collected over two years and smaller test part collected over two months. Each set is a time series that represents a temporal usage pattern of that particular machine. That temporal nature of the patterns was what caught our attention in the context of the Neurosolver.

We started with the use the data in their raw format by assigning each datum to a Neurosolver node. In that sense, each datum is a state of the system in the progression of states as specified by the given time series. The Neurosolver therefore learns the trajectory that corresponds to each training time series, and over time generalizes the trajectories to represent all time series by it adaptation rules. No states are hidden behind observations: states are the observations in this model.

Due to the large number of data points and the proximity of some of them, we also tried to cluster the data with several cluster sizes. For that, we approximated the k neighbor algorithm by one that is very straighforward in one dimension. Simply, we decided on an arbitrary number of clusters, and then recursively divided the data set into two subsets allocating the number of clusters for each of the two division according to the data density. An example of this process is shown in Figure 2. The desired number of clusters is 4, and the reader may notice that the left part of the set gets more clusters allocated due to a higher number of captured data points. The algorithm allows for a balanced distribution of data amongst a known number of clusters.

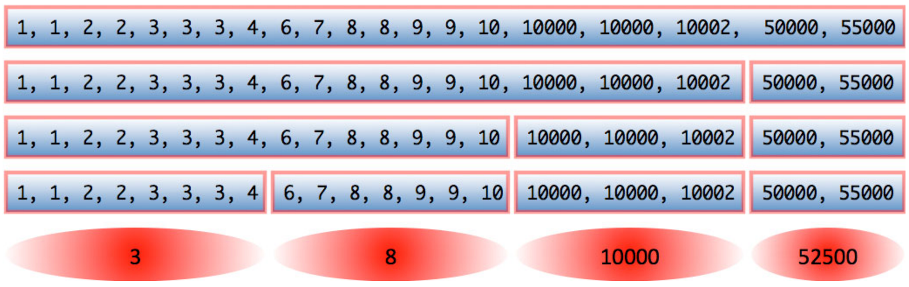


Fig. 2. An example of data clustering

A simpler approach to clustering is dividing the domain into a number of equal segments and then creating clusters based on the data membership in the clusters. However, the problem with this approach is that is does not take into consideration data distribution. Therfore, some clusters are not balanced: some might be empty, while others are overcrowded.

After the custering stage, we assigned the centers of the clusters to the Neurosolver's nodes. Subsequently, for each data point we activated the node that represented the cluster to which the point was classified. The predicted sequences were built also out of the numbers that corresponded to the centers of the clusters represented by the firing nodes.

4 Neurosolver Learning

4.1 Learning Rules

We used two types of learning in our experiments. The first follows the traditional incremental learning through gradient ascent (a.k.a gradient descent and hill-climbing) approaches (e.g., [13]) that are taken by many researchers in the neural networks community. The second, follows the stochastic scheme that was used in the original Neurosolver.

4.1.1 Incremental Learning

The Neurosolver learns by translating teaching samples representing state transitions into sequences of firing nodes corresponding to subsequent states in the samples. For each state transition, two connections are strengthened: one, in the direction of the transition, between the lower divisions of the two nodes, and another, in the opposite direction, between the upper divisions as shown in Figure 3. As we said earlier, the process is similar to the one employed in the forward-backward algorithm [11], although there is only one pass here.

In the incremental learning, we simply add a small value to the connection strength. We did not use any decay factor in the experiments reported here.

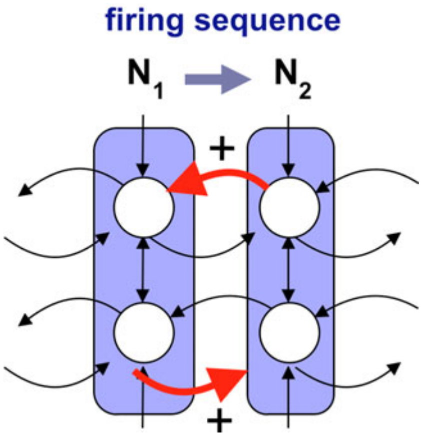


Fig. 3. Neurosolver learning rule

4.1.2 Statistical Learning

In the second approach, the strength of all inter-nodal connections is computed as a function of two probabilities: the probability that a firing source node will generate an action potential in this particular connection and the probability that the target node will fire upon receiving an action potential from the connection.

To compute the probabilities, each division and each connection collects statistics as shown in Figure 4. The number of transmissions of an action potential T_{out} is recorded for each connection. The total number of cases when a division positively influenced other nodes S_{out} is collected for each division. A positive influence means that an action potential sent from a division of a firing node to another node caused that node to fire in the next cycle. In addition, we also collect statistical data that relate to incoming signals. T_{in} is the number of times when an action potential transmitted over the connection contributed to the firing of the target node and is collected for each connection.

S_{in}, collected for each division, is the total number of times when any node positively influenced the node. With such statistical data, we can calculate the probability that an incoming action potential will indeed cause the target node to fire. The final formula that is used for computing the strength of a connection (shown in

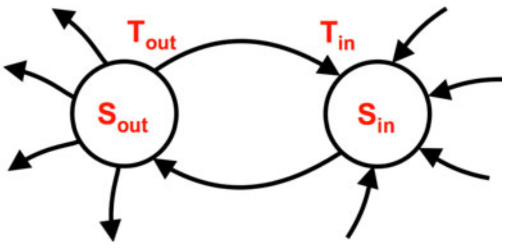


Fig. 4. Statistics collected for computation of the connection strength between nodes

Equation 1) is the likelihood that a firing source node will induce an action potential in the outgoing connection, multiplied by the likelihood that the target node will fire due to an incoming signal from the connection:

$$P = P_{out} \cdot P_{in} = (T_{out}/S_{out}) \cdot (T_{in}/ S_{in}) \quad (1)$$

4.2 Learning Sequences

As we already indicated, in the goal-oriented problem solving mode the function of the network of upper divisions is to spread the search activity along upper-to-upper connections. In the forecasting mode, the same network could be used to provide some guidance in forecasting as we indicate in the notes on the future directions of the research. However, in the experiments that we report in this paper, the network of the upper divisions is ignored. Instead, we focus on the functionality provided by the network built out of the lower divisions.

The purpose of the network composed of the lower divisions and their connections is to generate a sequence of output signals from firing nodes (along the connections shown in Figure 5). In the goal-oriented search mode, such a sequence corresponds to a specific path between the current state and the goal state, and—as stated earlier—can be considered a solution to the given problem.

In the forecasting mode, the node corresponding to the current state ("current observations") is activated through the thalamic input and allowed to fire. The activity from the firing node is transmitted to the nodes that are connected to the firing node through the efferent connections with non-zero strengths. Assuming a substantial learning sample, it is very likely that there is only one connection that is strongest, so the node that is connected through that connection is the winner of the contest for the highest activation level. The number that is the center of the cluster corresponding to that node is the predicted value. In the non-clustering tests, it is the datum that is associated with the node. Subsequently, the winning node is forced to fire next, and the process for selecting the next winner is repeated until no more predictions can be made.

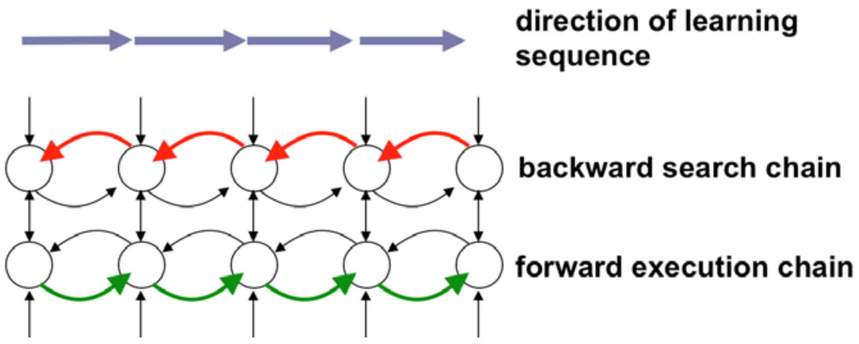


Fig. 5. The Neurosolver learn temporal patterns

4.3 Implementation Tweaks

4.3.1 Inhibition

As indicated earlier, to avoid oscillations, the firing node is inhibited for a number of computational cycles. The length of the inhibition determines the length of cycles that can be prevented. For example, an inhibition that lasts three cycles prevents cycles of three nodes. There are some negative implications here; for example, any pattern containing such a cycle cannot be predicted.

4.3.2 Higher-order Connections

Our initial implementation had first degree connections that link only to a direct predecessor of a node. We later enhanced our models with second degree connections, which provided a link to more distant predecessors in the Neurosolver's firing history. Adding connection degrees allows us to take into consideration a number of previously fired nodes when forecasting the next node to fire. In that respect, this approach is similar to Markov models [9]. We will analyze further enhancements in the section on future work.

5 Experiments

5.1 Quality Measure

To measure the quality of our predictions and to compare them with the benchmarks and submissions to the NN5 competition we used Symmetric Mean Absolute Percent Error (SMAPE) that was recommended by the NN5 organizers (Equation 2). The SMAPE calculates the symmetric absolute error in percent between the actuals *X* and the forecast *F* across all observations *t* of the test set of size *n* for each time series *s* with the following formula:

$$SMAPE_s = \frac{1}{n}\sum_{t=1}^{n} \frac{|X_t - F_t|}{(X_t - F_t)/2} * 100 \tag{2}$$

5.2 Results

We generated a substantial body of results running the NN5 data sets with numerous incarnations of the Neurosolver. We processed the data in the raw form, as well as pre-processed by clustering techniques as described earlier. We also tested the Neurosolver with the two learning algorithm: gradient ascent and stochastic.

In the following sections, we present an analysis of the Neurosolver's performance on some selected data. In the analysis, we compare several models and approaches that we used, and relate the results to the test data provided with the NN5 data sets. We conclude with a comparison with the benchmark predictions generated by non-neural methods provided by the organisers of the NN5 competition for reference [14].

5.2.1 Comparing Learning Models

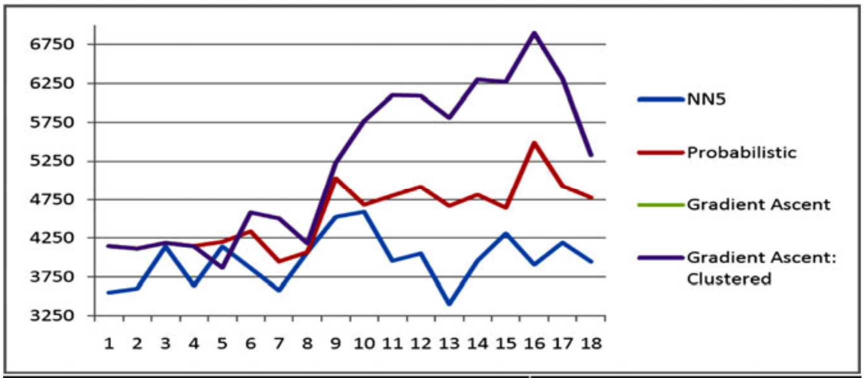
The graph in Figure 6 illustrates Neurosolver's predictions following presentation of one of the data points (4025) from the NN5 data sets. The four lines in the graph represent:

- the actual data provided by the NN5 competition (bottom at 16),
- our forecasted values for the stochastic model (top at 16), and
- forecasting made with two hill-climbing models (middle at 16).

The data in the table below the graph show the standard deviation between our results and the test data from the NN5 data sets.

The graph in Figure 6, illustrates how the Neurosolver behaves when the data is not shaped by clustering algorithms. We used a cluster size of one in the shown clustered gradient ascent, so a node is used per each value, making it virtually the same as an un-clustered model. Therefore, the Neurosolver generated the same forecasts for both the clustered and un-clustered gradient ascent models. The lines corresponding to the two gradient ascent models—middle at 16—are collapsed and displayed as a single purple line.

From the graph and the standard deviation between the forecasted values and the NN5 data (top at 16), we observe that the probabilistic model provides forecasts that are closer to the actual data (as provided with the NN5 sets) in terms of the standard deviation from the measured data.

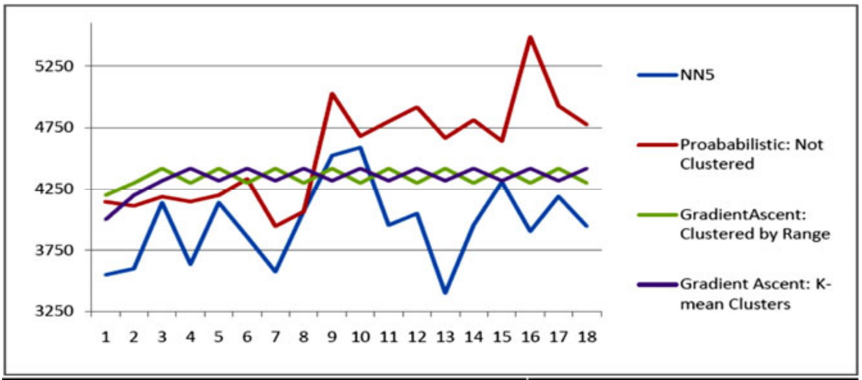


Algorithm (input 4025)	Standard Deviation from NN5
Probabilistic	423.832098 ✓
Gradient Ascent: Not Clustered	913.405457
Gradient Ascent: Clustered	913.405457

Fig. 6. Probabilistic vs. Gradient Ascent. NN5 competition: bottom at 16, stochastic model: top at 16, and two hill-climbing model: middle at 16.

5.2.2 The Clustering Factor

The graph in Figure 7 illustrates how the clustering algorithms affect the forecasting. The value used as the input to our forecaster is the same as before (4025). The



Algorithm (input 4025)	Standard Deviation from NN5
Probabilistic: Not Clustered	423.832098
Gradient Ascent: Clustered by Range (50)	254.011766 ✓
Gradient Ascent: K-mean Clusters (input/50)	257.527038

Fig. 7. Clustered vs. Unclustered. NN5 competition: bottom at 16, stochastic not clustered model: top at 16, and two hill-climbing models: middle at 16.

cluster-by-range gradient ascent model divides the input into 50 clusters. The k-means clustering algorithm divides the size of the input by 50, giving us 173 clusters for this particular data set. The significance of the clustering process can be seen in the change in our standard deviation for the gradient ascent model. The forecasted values from the gradient ascent models are now much closer to the actual data provided with the NN5 data sets.

The clustering algorithms provide an overall improvement in our results; however, we have encountered some cases in which the clustering algorithm increased our deviation from the actual values.

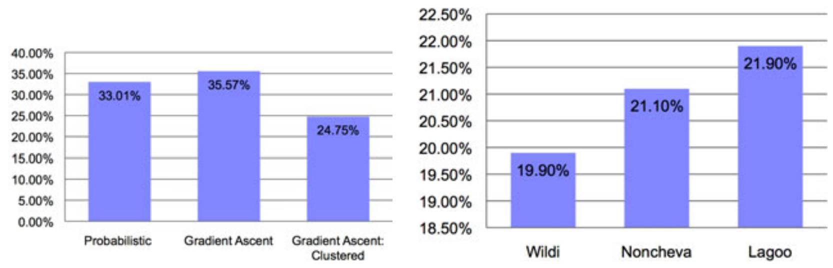
5.2.3 Comparing with the NN5 Submissions

The NN5 website provides a list of contest submissions and benchmark results. They use the SMAPE formula to calculate the quality of predictions generated by the competing and benchmark models. The best predictions that come from benchmark models are shown on the right side of Figure 8. No competing submission exceeded the performance of the benchmark models.

The left side of Figure 8 shows the performance of several models of the Neurosolver.

6 Conclusions

Currently our results are below the classical benchmarks on the NN5 website. However, we are not that much apart. We have found our current findings to be



Algorithm	SMAPE
Probabilistic	33.01%
Gradient Ascent	35.57%
Gradient Ascent: Clustered	24.75%

Algorithm	SMAPE
Wildi	19.90%
Noncheva	21.10%
Lagoo	21.90%

Fig. 8. Neurosolver vs. Benchmarks

promising and plan to apply a number of enhancements that we believe will improve the performance of the Neurosolver significantly.

Our initial implementation had first degree connections that link only to a direct predecessor of a node. We later enhanced our models with second degree connections, which provided a link to more distant predecessors in the Neurosolver's firing history. Adding 2nd-degree connections allowed us to take into consideration two (rather than one) previously fired nodes when forecasting the next node to fire. As we reported in [3], the addition of the second degree connection improved the prediction capabilities of the Neurosolver. In our new version of the Neurosolver, we are increasing the number of node predecessors to an arbitrary number, and call the new model n-order Neurosolver accordingly. We are going to use the new model to process the dataset posted for NN GC1 2010.

All the principles from the 1st-order Neurosolver are preserved, but the connectivity is enhanced as showed in Figure 9 that shows a connectivity of a single node in a 4-order Neurosolver. A connection from every predecessor of a node is subjected to the same learning rules as the connection from the direct predecessor. In the computing mode, the activity of every node is influenced by the activity of the n predecessors of the node.

We have concluded that a potential source for our deviation in forecasted values could be due to incomplete data in our learning set. In such cases the Neurosolver gets stuck. We will be looking into engineering some means to boost Neurosolver's activity to address such problems.

As we indicated, we use node inhibition to solve the problem with cycles that can lead to looping. At the same time, however, we may prevent generation of genuine cycles that may be present in the data sets. We believe that this is the main cause of the occasional inability to generate predictions at certain critical points. We are planning to look into developing a mechanism that would accommodate generating genuine cycles while still preventing endless loops.

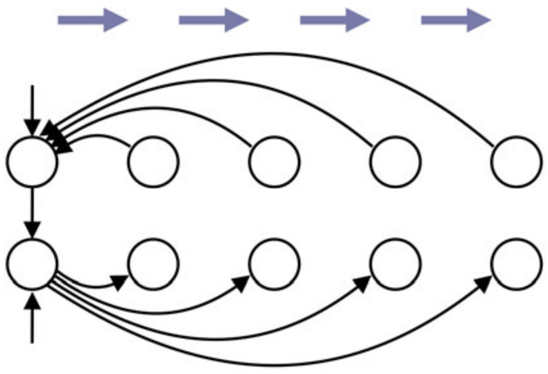


Fig. 9. The 4-order Neurosolver

Yet another venue that we are planning to pursue is a guided prediction. For that, we would employ the network of upper divisions. For example, we may want to limit the number of potential outcomes and present a number of goal states to the Neurosolver. Rather than spreading predictive activity only according to the strength of the connections, additional signals coming from the search network corresponding to the activation of the set of desired goals would also be taken into account.

Acknowledgements. The author would like to express gratitude to several students that were members of the research group operating in Spring 2009 semester under the umbrella of Faculty-Undergraduate Student Research Initiative supported by California State University Channel Islands. The following students provided great help with programming, testing, analyzing data and charting the results: Fahimeh Fakour, Douglas Holmes, Maximillian Kaufmann, and Nicholas Peters.

References

1. Bieszczad, A., Pagurek, B.: Neurosolver: Neuromorphic General Problem Solver. Information Sciences: An International Journal 105, 239–277 (1998)
2. Newell, A., Simon, H.A.: GPS: A program that simulates human thought. In: Feigenbaum, E.A., Feldman, J. (eds.) Computer and Thought. McGrawHill, New York (1963)
3. Burnod, Y.: An Adaptive Neural Network: The Cerebral Cortex. Masson, Paris (1988)
4. Laird, J.E., Newell, A., Rosenbloom, P.S.: SOAR: An architecture for General Intelligence. Artificial Intelligence 33, 1–64 (1987)
5. Nillson, N.J.: Principles of Artificial Intelligence. Tioga Publishing Company, Palo Alto (1980)
6. Collins, A.M., Loftus, E.F.: A spreading-activation theory of semantic processing. Psychological Review 82(6), 407–428 (1975)
7. Anderson, J.R.: A spreading activation theory of memory. Journal of Verbal Learning and Verbal Behavior 22, 261–295 (1983)
8. Doob, J.L.: Stochastic Processes. Wiley, Chichester (1953)
9. Meyn, S.P., Tweedie, R.L.: Markov Chains and Stochastic Stability, 2nd edn. Cambridge University Press, Cambridge (2009)

10. Markov, A.A.: An Example of Statistical Investigation of the Text Eugene Onegin Concerning the Connection of Samples in Chains. Translation by David Link. Science in Context 19(4), 591–600 (2006)
11. Baum, L.E., Petrie, T.: Statistical Inference for Probabilistic Functions of Finite State Markov Chains. Annals of Mathematical Statistics 37(6), 1554–1563 (1966)
12. Bieszczad, A.: Exploring Neurosolver's Forecasting Capabilities. In: Proceedings of the 28th International Symposium on Forecasting, Nice, France, June 22-25 (2008)
13. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn., pp. 122–125. Prentice Hall, Upper Saddle River (2010)
14. NN5, `http://www.neural-forecasting-competition.com/`

Author Index